NASA/CR—2006-214225

Enhancement of the Feature Extraction Capability in Global Damage Detection Using Wavelet Theory

Atef F. Saleeb and Gopi Krishna Ponnaluru
University of Akron, Akron, Ohio

May 2006

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 301–621–0134

- Telephone the NASA STI Help Desk at 301–621–0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076–1320

NASA/CR—2006-214225

Enhancement of the Feature Extraction Capability in Global Damage Detection Using Wavelet Theory

Atef F. Saleeb and Gopi Krishna Ponnaluru
University of Akron, Akron, Ohio

Prepared under Cooperative Agreement NCC3–808

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

May 2006

Acknowledgments

The financial support provided for this work by NASA Glenn Research Center under cooperative agreement NCC3–808 to the University of Akron is gratefully acknowledged. We would like to thank Dr. Stevem M. Arnold (Life Prediction Branch) for his extensive co-operation and help. Our special thanks to Dr. Thomas Wilt for his valuable assistance throughout this study.

Level of Review: This material has been technically reviewed by NASA expert reviewer(s).

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Available electronically at http://gltrs.grc.nasa.gov

Enhancement of the Feature Extraction Capability in Global Damage Detection Using Wavelet Theory

Atef F. Saleeb and Gopi Krishna Ponnaluru
University of Akron
Akron, Ohio 44325

Summary

The main objective of this study is to assess the specific capabilities of the defect energy parameter technique for global damage detection developed by Saleeb and coworkers. The feature extraction is the most important capability in any damage detection technique. Features are any parameters extracted from the processed measurement data in order to enhance damage detection. The damage feature extraction capability was studied extensively by analyzing various simulation results. The practical significance in structural health monitoring is that the detection at early stages of small-size defects is always desirable. The amount of changes in the structure's response due to these small defects was determined to show the needed level of accuracy in the experimental methods.

The arrangement of fine/extensive sensor network to measure required data for the detection is an "unlimited" ability, but there is a difficulty to place extensive number of sensors on a structure. Therefore, an investigation was conducted using the measurements of coarse sensor network. The white and the pink noises, which cover most of the frequency ranges that are typically encountered in the many measuring devices used (e.g., accelerometers, strain gauges, etc.) are added to the displacements to investigate the effect of noisy measurements in the detection technique. The noisy displacements and the noisy damage parameter values are used to study the signal feature reconstruction using wavelets. The enhancement of the feature extraction capability was successfully achieved by the wavelet theory.

Chapter I
Introduction

1.1 General

Engineering structures are considered valuable assets to a nation due to their cost, complexity, and importance. These structures may deteriorate or damage during their service lives due to various reasons, that is, environment, loads, etc. The interest in the ability to monitor a structure and detect the damage at the earliest possible stage is common throughout the civil, mechanical, and aerospace engineering fields. Undetected damage may cause failure leading to loss of life and property. If the "state of health" of an important structure is constantly monitored, its maintenance cost can be kept to a reasonable minimum. The cost of repair is obviously less than that required to reconstruct the whole structure. As a result, structural health monitoring has received considerable attention in the literature.

The increase in research activity regarding damage detection is the result of many factors. These factors can be generally categorized as spectacular failures resulting in loss of life, economic concerns, and recent technical advancements. Advancements in sensors and in the finite-element method have contributed to recent improvements in damage detection. Additional factors that have contributed to these improvements are the adaptation and advancements in experimental techniques such as modal testing and development of linear and nonlinear system identification methods.

Damage is defined as changes introduced into a structure, either intentional or unintentional, which adversely affect the current or future performance of that structure. Damage may also be defined as any deviation in the structures original geometry or material properties that may cause undesirable stresses, displacements, or vibrations on the structure. Damage, such as a crack, in a structure reduces its stiffness. Similarly, the damping ratio will increase when damage progresses in the structure. The natural frequencies and the mode shapes are directly related to the stiffness of the structure. The loss of stiffness causes a drop in natural frequencies and a change in mode shapes. During the last decade, the use of dynamic system parameters such as natural frequencies, damping ratios, and mode shapes to detect damage has been studied intensively and many damage-detection techniques have been developed.

Even though there has been significant research in the field of nondestructive damage identification, several factors remain to be tested to judge the potential of any damage-detection technique before it becomes a routine activity. Factors such as the limitation of measuring only a few modes in the experiments particularly in large structures, the limitation in the arrangement of an extensive sensor network, the experimental errors in the measurements, and the feature extraction capability to determine the location and the severity of the damage are to be investigated. Along with the above investigations, feature reconstruction methods used to extract the required signal from actual noisy data measurements need to be studied so as to enhance the feature extraction capabilities of the global damage detection techniques.

1.2 Problem Statement

Extensive research work over the past years using concepts of structural health monitoring have been directed towards the development of many damage-detection techniques. For the success of any technique, the following items should be studied:

(1) The ability of the damage-detection scheme to withstand noisy measurements
(2) Practicality of the density of the sensor network needed
(3) Feature extraction and representation
(4) Feature reconstruction with significant amounts of data/measurement noise

In the present study, the focus will be on the detail assessment of the four items listed above in connection with the specific defect energy parameter technique developed recently for global damage detection by Saleeb, et al. (refs. 67, 68, 80, and 85). In particular, extensive utilization of wavelet theories and graphical toolkits are made for the studies conducted under items (3) and (4) above.

1.3 Objective of the Study

The objective of this study is the assessment of the capabilities of the global damage detection technique developed by Saleeb and coworkers. This study presents the work of damage identification on plates when subjected to out-of-plane vibrations for a variety of cases. The assessment includes the validation of the technique by considering different types of failure/damage modes; the investigation on the density of the sensor network, where measurements on coarse sensor networks have particular importance; determination of the noise level that can be withstood by the detection technique; conduction of the false alarm tests; and the investigation of the damage feature extraction and representation for the above cases. An additional goal of this study is to reconstruct the damage feature from the noisy measurements by the utilization of the denoising tools, that is, wavelet transforms.

1.4 Outline

The introductory chapter is followed by background and review on literature in the areas of damage detection, data noise simulations, and the feature reconstruction using wavelet analysis in chapter II. This will follow the theoretical development on the proposed damage technique and wavelet theory in chapter III. Chapter IV describes the computational tools used to complete this work. The results and simulation case studies are presented in chapter V. Finally, a summary and conclusions are presented in chapter VI.

Chapter II
Background and Literature Review

2.1 Background

Damage detection is a challenging problem that is under vigorous investigation by numerous research groups using a variety of analytical and experimental techniques. A significant amount of research work over the years has dealt with general topic of structural health monitoring. Health monitoring techniques may be classified as global or local. Global methods attempt to simultaneously assess the condition of the whole structure, whereas local methods provide information about a relatively small region of the system by using local measurements. Clearly, these approaches are complementary to each other, with the choice of methods being dependent on the scope of the problem at hand and nature of the sensor network on the structure. The damage detection procedure depends on the level of damage and deterioration of concern, the nature of the instruments, the spatial resolution of the sensors, the degree of the measurement noise, the configuration of topology of the test structures, complexity of the detection scheme, and the depth of knowledge concerning the failure modes of the structure.

Another confounding factor is the fact that damage typically is a local phenomenon. Local response is captured by higher frequency modes whereas lower frequency modes tend to capture the global response of the structure and are less sensitive to local changes in a structure. From a testing standpoint it is more difficult to excite the higher frequency response of the structure, as more energy is required to produce measurable response at these higher frequencies than at the lower frequencies. These factors coupled with the loss of information resulting from the necessary reduction of time-history measurements to modal properties add difficulties to the process of vibration-based damage identification.

The Non Destructive Evaluation (NDE) methods for detection of damage in structural systems have received increasing attention in the recent past for different classes of structural components using the signature analysis of the system response. While there are many approaches that have been investigated, or still being developed for signature based NDE of structure, the class of health-monitoring approaches that do not require detailed knowledge of the vulnerable parts of the structure, or of the failure mode of the structure, have a significant advantage in that they have the potential to cope with unforeseen failure patterns. Classes of NDE schemes that are less sensitive to the effects of initial assumptions are certainly of significant advantage. Considering the details of the NDE technique, focus is placed on the response signature-analysis, providing a global approach for fault diagnosis, as opposed to other local methods that require one to look directly at the location of suspected defects.

The effect of damage on a structure can be classified as linear or nonlinear. A linear damage situation is defined as the case when the initially linear-elastic structure remains linear-elastic after damage. The changes in modal properties are a result of changes in the geometry and/or the material properties of the structure, but the structural response can still be modeled using linear equations of motion. Linear methods can be further classified as model-based and non-model-based. Model-based methods assume that the monitored structure responds in some predetermined manner that can be accurately discretized by finite element analysis, such as the response described by Euler-Bernoulli theory.

Nonlinear damage is defined as the case when the initially linear-elastic structure behaves in a nonlinear manner after the damage has been introduced. One example of nonlinear damage is the formation of a fatigue crack that subsequently opens and closes under the normal operating vibration environment. Other examples include loose connections that rattle and nonlinear material behavior such as that exhibited by polymers. The majority of the studies reported in the technical literature address only the problem of linear damage detection.

Another classification system for damage-identification methods defines four levels of damage identification, as follows:

- Level 1: Determination of the presence of damage in the structure.
- Level 2: Level 1 plus determination of the geometric location of the damage.

- Level 3: Level 2 plus quantification of the severity of the damage.
- Level 4: Level 3 plus prediction of the remaining service life of the structure.

Level 1 is the most critical, whereas level 2 and level 3 would generally require additional more detailed measurements. Level 4 prediction is generally associated with fields of fracture mechanics, fatigue-life analysis, or structural design assessment. There are two alternative approaches for mathematical representation and implementation for these global detection techniques, i.e., system-identification and direct post processing of measurement data. The approaches mainly differ in the amount and type of information sets utilized. In particular, the system-identification approach is typically based on a complete analytical model that is fitted to the measured response

Within the framework of system identification techniques, there is a distinction between methods that are used for continuous monitoring of structure performance and methods that are applicable to the detection of damage caused by extreme events. As an example, a system that uses continuous or intermittent accelerometer measurements from sensors mounted permanently to a bridge is different in terms of instrumentation and data acquisition requirements from a system that does not acquire data except during and immediately following an earthquake or a hurricane. It should be noted that primary distinction between these situations has to do with the sensors and data acquisition system requirements. Typically, the same types of analytical techniques can be applied to the data to determine the integrity of the structure.

In addition to the intense computational demands, another main disadvantage of the system identification method is the need to treat the 'inherent nonuniqueness' caused by incompleteness of measured data with noise. Most recently, artificial neural networking has been used as an attempt to remedy some of these problems, but they still remain computationally intensive.

On the other hand the direct detection scheme requires a key ingredient for the selection of the appropriate 'measure' or damage parameter, which is sufficiently sensitive to any slight perturbation in system properties. The whole implementation simply reduces to processing of raw experimental data, the objective being pattern-recognition, using the notion of comprised processed signatures, i.e., present (damage) vs. the base/reference (intact) states. Sharper resolution will be obtained if there are more distinct differences between two signatures. The presence of noisy data and consistency of distinctive patterns under different excitations (i.e., different vibration modes or different static load intensities) reflects on the robustness of NDE technique. It is this type of direct global detection methodology that is used in the later parts for the parametric and the assessment studies performed in this report.

NDE technique should be robust enough to withstand certain level of unavoidable noise in practical measurements. If the technique is excessively sensitive to the noise levels then the effect of "true" damage will be completely "overshadowed" by the noise. With an eye towards reducing noise intensity, there has been development of different types of filters. The important factor in these so-called "denoising" schemes in the damage detection is retaining the significant, damage-produced features, of the signals. In the last two decades there has been extensive research going on the use of wavelet theory for this purpose. One of the applications of the wavelet theory is denoising. Wavelet theory is a useful tool to reduce the intensity of the noise in damage identification technique. Indeed, it is one of the main objectives in the present study to demonstrate the practical usefulness of applying the wavelet theories as a "denoising" tool in conjunction with a robust (global/direct) detection technique.

With the above background in mind, we proceed next to summarize the pertinent literature. For convenience in the presentations, these are grouped into three separate parts; i.e., (a) Global Detection, (b) Data Noise Simulations/Filtering schemes as well as (c) applications of the wavelet theories in Denoising/Signal – Feature reconstructions.

2.2 Literature Review—An Overview

The term damage refers to degradation or a failure of material. It can originate from diverse phenomena such as oxidation, carbonation, mechanical work, or any type or disintegration or weakening from aging or mechanical process. Within the framework of damage mechanics, only that which causes the loss of area, associated with change in local material properties such as Young's modulus E, moment of inertia I, stiffness and flexibility and energy dissipation, is considered. When such changes occur there is a change of the entire physical system. This change leads to a change in vibration characteristics in physical space and also in material space.

In physical space, when abrupt reductions in the cross section of a beam are considered, the properties have been changed, especially for the relationships between geometry and the centerline deflection curves. For free vibration behavior, such a beam results in a noticeable error in the natural frequencies because of the overestimation of the bending stiffness. It was found that resonant frequency and vibration amplitudes were considerably affected by the presence of cracks. Changes in the energy with respect to the damage are also important. Damage to engineering materials essentially results in a decrease of the free energy stored in a body with consequent degradation of the material stiffness.

In material space, damage creates heterogeneity in a homogenous body, with consequent reduction of material properties including Young's modulus, stiffness and moment of inertia.

2.3 Literature Review—Detection Methods of System Identification Type

Difficult challenges in formulating damage parameters possessing all these and other desirable attributes (e.g., applicability to different materials, multiple damage sites, various support condition, different vibration modes, etc.) are indicated by the numerous proposals made over the years (e.g., natural frequencies, mode shapes, influence flexibility coefficients, strain mode shape, curved mode shapes, as well as ratios, differences, fractions obtained form them). Basically, the performance of these measures was found to be heavily problem-dependent, with several conflicting conclusions often reached, when using the same measure under different conditions.

Damage identification methods are based on the modification of structural model matrices such as mass, stiffness, and damping to reproduce as closely as possible the measured static or dynamic response data. These methods solve for the updated matrices (or perturbations to the nominal model that produce the updated matrices) by forming a constrained optimization problem based on the structural equations of motion, the nominal model, and the measured data. Comparisons of the updated matrices to the original correlated matrices provide an indication of damage and can be used to quantify the location and the extent of the damage. The methods use a common basic set of equations, and the differences in the various algorithms can be classified as follows:

1. Objective function to be minimized
2. Constraints placed on the problem
3. Numerical scheme used to implement the optimization.

Chen and Garba (ref. 1) presented a method for minimizing the norm of the model property perturbations with a zero modal force error constraint. They also enforce a connectivity constraint to impose a known set of load paths onto the allowable perturbations. The updates are thus obtained at the element parameter level, rather than at the matrix level. This method is demonstrated on a truss FEM.

McGowan, et al.(ref. 2) report ongoing research that examines stiffness matrix adjustment algorithms for application to damage identification. Based on the measured mode shape information from sensor locations that are typically fewer than the DOF in an analytical model, mode shape expansion algorithms are employed to extrapolate the measured mode shapes such that they can be compared with analytical model results. These results are used to update the stiffness matrix while maintaining the connectivity and the sparsity of the original matrix.

Smith (ref. 3) presented an iterative approach to the optimal update problem that enforces the sparsity of the matrix at each iteration cycle. Multiplying each entry in the stiffness update by either one or zero enforces the sparsity pattern. Kim and Bartkowicz (ref. 4) investigated damage detection capabilities with respect to various matrix update methods, model reduction methods, mode shape expansion methods, number of damaged elements, number of sensors, number of modes, and levels of noise. The authors developed a hybrid model reduction/eigenvector expansion approach to match the order of the undamaged analytical model and the damage test mode shapes in the matrix update. They also introduced a more realistic noise level into frequencies and mode shapes for numerical simulation. From both numerical and experimental studies, the authors showed that the number of sensors is the most critical parameter for damage detection, followed by the number of measured modes. Lindner, et al. (ref. 5) presented an optical update technique that formulates an over determined system for a set of damage parameters representing reductions in the extensional stiffness values for each member. The value represents the amount of stiffness reduction in that member.

Lie (ref. 6) presented an optimal update technique for computing the elemental stiffness and mass parameters for a truss structure from measured modal frequencies and mode shapes. The method minimizes the norm of the modal force error. The author demonstrates that if sufficient modal data is variable, the elemental properties can be directly computed using the measured modal frequencies, measured mode shapes, and two matrices which represent the elemental orientations in space and the global connectivity of the truss. In this case, the solution for the elemental properties is shown to be unique and globally minimal. The method is used to locate a damaged member in a FEM of a truss using the first four measured modes in sets of three at a time.

Zimmrman and Kaouk (ref. 7) presented the basic minimum rank perturbation theory (MRPT) algorithm. A nonzero entry in the damage vector is interpreted as an indication of the location of the damage. The resulting perturbation has the same rank as the number of modes used to compute the modal force error. It is demonstrated that the MRPT algorithm preserves the rigid body modes of the structure and the effects of measurement and expansion errors in the mode shapes are demonstrated and discussed.

Kaouk and Zimmerman (ref. 8) further developed this algorithm and demonstrated how perturbations to two of the property matrices can be estimated simultaneously by using complex conjugates of the model force error equation. The method is demonstrated numerically for a truss with assumed proportional damping. Also, the technique is used experimentally to locate a lumped mass attached to a cantilevered beam.

Kaouk and Zimmerman (ref. 9) extended the MRPT algorithm to estimate mass, stiffness, and proportional damping perturbation matrices simultaneously. The computation of these individual perturbation matrices is accomplished by exploiting the cross-orthogonally conditions of the measured mode shapes with respect to the damage property matrices. The authors examined the results by computing a cumulative damage vector.

Kaouk and Zimmerman (ref. 10) presented a technique that can be used to implement the MRPT algorithm with no original FEM. The technique involves using a baseline data set to correlate an assumed mass and stiffness matrix, so that the resulting updates can be used as the undamaged property matrices.

Zimmerman and Simmermacher (refs. 11 and 12) computed the stiffness perturbation resulting from multiple static load and vibration tests. This technique is proposed partially as a method for circumventing the mismatch in the number of models between test and FEM. They applied this technique to a FEM of a structure similar to a NASA test article. They also presented two techniques for overcoming the rank deficiency that exists in the residual vectors when the results of one static or modal test are linear combinations of the results of previous tests.

Ricles (ref. 13) presented a methodology for sensitivity-based matrix update, which take into account variations in system mass and stiffness, center of mass locations, changes in natural frequency and mode shapes, and statistical confidence factors for structural parameters and experimental instrumentation. The method uses a hybrid analytical/experimental sensitivity matrix, where the modal parameter sensitivities are computed from the experimental data, and the matrix sensitivities are computed from the analytical

model. This method is further developed and applied to more numerical examples by Ricles and Kosmatka.

Sanayei and Onipede (ref. 14) presented a technique for updating the stiffness parameters of a FEM using results of a static load-displacements test. A sensitivity-based, element-level parameter update scheme is used to minimize the error between the applied forces and forces produced by applying the measured displacements to the model stiffness matrix. The sensitivity matrix is computed analytically. The structural DOF are partitioned such that the locations of the applied loads and the locations of the measured displacements are completely independent. The technique is demonstrated on two FEM examples.

In a related paper, Sanayei, et al. (ref. 15) examined the sensitivity of the previous algorithm to noisy measurements. The influence of the selected measurement DOF set on the error in the identified parameters is studied. A heuristic method is proposed that recursively eliminates the measurement DOF that the elemental stiffness parameters are the most sensitive. In this manner, the full FEM DOF set is reduced to a manageable size while preserving the ability to identify the structural stiffness parameters.

Hemez and Farhat (ref. 16) presented a sensitivity-based matrix update procedure that formulates the sensitivities at the element level. This has the advantage of being computationally more efficient than forming the sensitivities at the global matrix level. It also allows the analysis to "focus" on damage in specific members.

Zimmer and Kaouk (ref. 17) implemented an eigenstructure assignment technique for damage detection. They included algorithms to improve the assignability of the mode shapes and preserve sparsity in the updated model. They applied their technique to the identification of the elastic modulus of a cantilevered beam.

Linder and Goff (ref. 18) defined damage coefficients for each structure member. They then used an eigenstructure assignment technique to solve for the damage coefficient for each member. They applied this technique to detect simulated damage in a 10-bay truss FEM.

Schulz, et al. (ref. 19) presented a technique similar to eigenstructure assignment known as "FRF assignment". The authors formulated the problem as a liner solution for element-level stiffness and mass perturbation factors. They pointed out that using FRF measurements directly to solve the problem is more straightforward than extracting mode shapes. They used measured mobility functions (FRFs from velocity measurements) to obtain higher numerical accuracy, since the velocity response is flatter over the entire spectrum than either the displacement or acceleration response. The technique is applied to an FEM of a bridge structure.

Baruh and Ratan (ref. 20) used the residual modal force as an indicator of damage location. They separated the residual modal force into the effects of identification error in the measurements, modeling error in the original structural model, and modal force error resulting from structural damage. They examined the sensitivity of the damage location solution to errors in the original structural model and to inaccuracies in the modal identification procedure.

Kim and Bartowicz (ref. 21) and Kim, et al.(ref. 22) presented a two-step damage-detection procedure for large structures with limited instrumentation. The first step uses optimal matrix update to identify the region of the structure where damage has occurred. The second step is a sensitivity-based method, which locates the specific structural element where damage has occurred. The first advantage of this approach lies in the computational efficiency of the optimal update method in locating which structural parameters are potentially erroneous. The second advantage lies in the small number of parameters updatation by the sensitivity-based technique.

Li and Smith (ref. 23) presented a hybrid model update technique for damage identification that uses a combination of the sensitivity and optimal-update approaches. This method constrains the stiffness matrix perturbation to preserve the connectivity of the FEM, and the solution minimizes the magnitude of the vector of perturbations to the elemental stiffness parameters.

Hung-Liang, et al. (ref. 24) presented a nondestructive evaluation method to identify the structural stiffness of ceramic candle filters. All filters were subjected to an excitation force, and the response was picked up by an accelerometer in a free-free boundary condition. The frequency response function and

vibration mode shapes of each filter were evaluated. The results are also estimations of the overall bending stiffness values for four different types of candle filters. The used filters showed stiffness degradation. The location and the size of the damaged section were identified using the measured modal strain energy.

Dynamic bending stiffness was used by J. Maeck et al. (ref. 25) to detect damage. They discussed different techniques and derived from experimentally determined modal characteristics of a reinforced concrete beam from its dynamic bending stiffness. The degradation of stiffness, due to cracking of the reinforced concrete, gives information on the position and severity of the damage that has occurred.

Yuen (ref. 26) presented a systematic study of the relationship between damage location and size, and the change in the eigenvalues and eigenvectors of a cantilever beam. Damage was modeled as a modulus reduction in an element of the beam. The eigen parameters were studied, i.e., translation and rotation. Both showed a sudden change at the damage region along the beam coordinate. However, the rotation eigen parameter was not detected for higher modes.

Ren and De Roeck (refs. 27 and 28) proposed a damage identification technique based on a change in frequencies and mode shapes of vibration, for predicting damage location and severity. The method is applied at an element level with a conventional finite element model. The element damage equations have been established through the eigen-value equations that characterize the dynamic behavior. The influence of noise was also shown and they verified their method by a number of damage scenarios for simulated beams and found the exact location and the severity of damage. They demonstrated that multiplying the damage eigen-value equations with the undamaged or damaged mode shapes provides more equations and guarantees the damage localization.

Stubbs and Topole (ref. 29) proposed a formulation that localizes and determines the size changes in the stiffness of the structure. Generally such changes are a reduction in stiffness and are associated with some type of structural damage. Serious damage will change the stiffness locally and globally. Thus a reduction in stiffness is generally interpreted as damage. However, reductions in stiffness do not necessarily relate to damage. Therefore, the algorithm is a conservative method to determine potential locations of damage.

Gawronski and Sawicki (ref. 30) used modal and sensor norms to determine damage locations in flexible structures. It provided information about the impact of the damage on the natural modes of the damage structures. As the norm is determined from the system natural frequencies, modal damping ratios, and the input and output gains; they depend neither on the input time history nor the actual system deformation.

From fracture mechanics concepts, Rizo et al.(ref. 31) developed a spectral method to identify cracks in various structures. Crack depth was related to the change of natural frequencies, but this method lacks accuracy for small cracks, and it is not quite clear that this approach can be generalized to complex structures.

F. Vestroni and D. Capecchi (ref. 32) found damage by frequency measurement. A linear behavior was assumed, before and after the damage. The method was described and used when frequencies are the observed quantities. The procedure is generalized by assuming finite-element interpretative models and an automatic algorithm of modal updating, which is used to determine the best stiffness distribution for an assigned location of damage. A minimum amount of frequencies is necessary to obtain a unique solution. This is important, because the problem often over determined. Quantity of measured data is important to reach an acceptable solution.

2.4 Literature Review—Detection Methods of Direct/Pattern Recognition Global Type

One approach to structural health monitoring is to look for changes in a “signature” of the structure that is related to its dynamic characteristics. The observed changes in the structure, for example modal parameters, are compared to a database of possible changes and the most likely change is selected for detecting damage and locating its position. It is first noted that for beams, plates, and shells there is a

direct relationship between curvature and bending strain. Some researchers discuss the practical issues of measuring strain directly or computing it from displacements or accelerations.

Stubbs, et al.(ref. 33) presented a method based on the decrease in modal strain energy between two structural DOF, as defined by the curvature of the measured mode shapes. Topole and Stubbs (refs. 34 and 35) examined the feasibility of using a limited set of modal parameters for structural damage detection. They studied several dynamic parameters for damage detection using full scale modal testing. A probable failure due to a large fatigue crack was simulated by unfastening a set of high-strength bolts in a splice connection of a steel highway bridge. Experimental modal testing was performed for intact case as well as the cracked case. Results indicate the presence of detectable changes in some of the response data to a simulated physical failure. Non-parametric information, i.e., time records, frequency spectra, transfer functions as well as parametric information, i.e., modal frequencies and mode shapes had been examined. The Modal Assurance Criteria (MAC) and modal frequencies can detect the damage in higher modes; otherwise the modal frequencies were not sensitive. In more recent publications, Stubbs and Kim (ref. 36) examined the feasibility of localizing damage using this technique without baseline modal parameters.

Pandey, et al.(ref. 37) developed curvature-mode shapes in which the absolute change is located in the region of damage, and hence can be used to detect damage in a structure. The changes in the curvature mode shape increases with increasing damage. The difference in modal curvature between the intact and the damaged beam showed not only a high peak at the fault position, but also some small peaks at different undamaged locations for higher modes. This can cause confusion to the analyst in a practical application in which one does not know in advance the location of faults. Also, it is not sensitive for small damages.

Chance, et al. (ref. 38) found that numerically calculating curvature from mode shapes resulted in unacceptable errors. They used measured strains instead to measure curvature directly, which dramatically improved results.

One feasible measurement property is change of energy with respect to material damage. Damage to engineering materials results in a decrease of the material stiffness. DisPasquqle, et al. (ref. 39) found that the parameter based global damage indices could be related to locate damage variables through operations that average over the body volume.

Stubbs and Osegueda (refs. 40 and 41) evaluated non-destructive damage detection. The damage was modeled by reduced modulus in the element. A finite element model was used to establish the sensitivity matrix. Laboratory experiments were performed for supporting the concept. As far as the damage location is concerned, false predictions still occurred. Moreover, from the experimental results, small damage levels cannot be detected.

West (ref. 42) presented what is possibly the first systematic use of mode shape information for the location of structural damage without the use of a prior FEM. The author uses the Modal Assurance Criteria (MAC) to determine the level of correlation between modes from the test of an undamaged Space Shuttle Orbiter body flap and the modes from the test of the flap after it has been exposed to acoustic loading. The mode shapes are partitioned using various schemes, and the change in MAC across the different partitioning techniques is used to localize the structural damage.

Mayes (ref. 43) presented a method for model error localization based on mode shape changes known as structural translational and rotational error checking (STRECH). By taking ratio of relative modal displacements, STRECH assess the accuracy of the structural stiffness between two different structural degrees of freedom (DOF). STRECH can be applied to compare the results of test with an original FEM or to compare the results of two tests.

Ratcliffe (ref. 44) presented a technique for locating damage in a beam that uses a finite difference approximation of Laplacian operator on mode shape data. Cobb and Liebst (ref. 45) presented a method for prioritizing sensor locations for structural damage identification based on an eigenvector sensitivity analysis.

Modal curvature method was used by Wahab, et al.(ref. 46) to detect damage in a pre- stressed concrete bridge. To establishment their method they used simulated data from simply supported and

continuous beams containing damage parts at different locations. A damage indicator called a "curvature damage factor" was introduced which is the difference in curvature mode shapes for all modes and could be summarized by one number for each measured point. For several damage locations in the structure, all modes should be carefully examined. The lower modes are, in general, more accurate than the higher modes. When more than one fault exists in the structure, it is not possible to locate damage in all positions from the result of only one mode.

Wavelet-based approach for Structural Health Monitoring (SHM) and damage was used by Z. Hou, et al. (ref. 47). The method is applied to simulated data generated from a simple structural model subjected to a harmonic excitation. Spikes in details of the wavelet decomposition may detect changes in system stiffness, and the locations of these spikes indicate the moment when structural damage occurred.

J.C. Hong, et al. (ref. 48) applied wavelet transforms for the damage identification in a structural member. They showed the effectiveness of the wavelet transforms by means of its capability to estimate the Lipshitz exponent. A model beam which has a defect represented by an abrupt thickness change is taken for the study. They derived bound of the Lipschitz exponent using the simplest Euler beam theory. Based on the estimated Lipschitz exponent, they suggested an optimal wavelet for the estimation of the Lipschitz exponent. The Continuous Wavelet Transform (CWT) by a Mexican hat wavelet having two vanishing moments is utilized. CWT is applied on the mode shape of the damaged beam. The modulus maximum lying inside of the cone of influence is clearly seen in the contour plot of CWT. It is also able to detect the location of the damage.

Y.J. Yan, et al.(ref. 49) presented structural damage feature index using dynamic response of the structure. They used honeycomb sandwich plate with free-free boundary conditions. Dynamic responses are measured using piezo-patch sensor. Wavelet transforms are utilized to calculate the damage index. Dynamic response signal of the plate is decomposed into 5^{th} layer of wavelet transform, and 16 sub-signals are obtained. The relative energy variation in each sub-signal is calculated. This approach is evaluated by comparing simulated and experimental data. The error between simulated and experimental data energy variation is less than 10 percent with a crack length more than 3 percent of the length of the plate. This method can able to tell whether the damage is exist or not, but it can not able to tell the location and severity of the damage.

2.5 Literature Review - Data Noise Simulation and Filtering Schemes

When the measured data is noise free, an element can be viewed as damaged if its estimated parameter is different from the baseline value. This simple assessment is complicated by the presence of measurement noise. The noise in the measurements would cause the estimated parameter for an element to be different from the baseline value even if there is no damage at all.

Ikumasa Yoshida and Tadanobu Sato (ref. 50) suggested from the standpoint of damage detection, non-Gaussian non-white noises might be preferable, because the damage tends to be concentrated on a specific part of a structure. From this consideration, exclusive noise which is non-Gaussian and non-white is proposed. They explained from observations of actual damages, proper process noise for damage detection should have following nature, i) concentration in space, ii) correlation in time domain. Proposed exclusive noise has both nature i) and ii). The exclusive noise is given to only one element at each step, so that it is consistent with the damage concentration nature. At the next step, the element is given the noise is selected with probability *Ps*, otherwise an element is selected randomly. Consequently, the noise has correlation in time domain.

Taeho Charles Jo (ref. 51) presented on MLP training with noise optimized by genetic algorithm. He explained the noise addition to the training patterns as follows. The input vector x_i is represented $x_i = [x_{i1}, x_{i2}, \ldots\ldots, x_{im}]$. The dimension of input vector is *m.* The noise injected into the input vector x_i of training pattern and ε_i is represented as $\varepsilon_i = [\varepsilon_{i1,}\varepsilon_{i2,}\ldots\ldots\ldots,\varepsilon_{im}]$. The element ε_{ik} of ε_i is the random number

based on Gaussian distribution. Therefore, the noise is injected into the input vector like $x'_i = [x_{i1} + \varepsilon_{i1,} x_{i2} + \varepsilon_{i2,} ....., x_{im} + \varepsilon_{im}]$. In this case, the input vector with noise is x'_i.

Xu and Liu (ref. 52) developed a combined optimization technique of using an improved μ GA and local optimizer to solve the optimization problem so as to obtain the flaw parameters defining flaw configuration. To investigate the effect of noise, they introduced white noise $\Gamma(x)$ intentionally in the simulated measurements for the surface displacement responses. $\Gamma(x)$ is described as $\sum_{i=1}^{n} \Gamma(x_i) = 0$, $\sum_{i=1}^{n} \Gamma(x_i)\Gamma(x_i - \tau) = 2D\delta(\tau)$ and $D = \eta\left(\frac{1}{n-1}\sum_{i=1}^{n} x_i^2\right)^{1/2}$, where η is the noise amplitude. It can be seen that the improved μ GA can bear relatively weak noise disturbances of $\eta = 5$ percent. To effectively suppress the effect of noise, they suggested to filter the noisy displacement responses before they are used in the objective function. After filtering the noisy data they can able to go up to $\eta = 15$ percent.

Cacciola, Impollonia, and Muscolino (ref. 53) adopted Monte Carlo Method for the crake detection in a damaged beam. They explained the analytical model of the cracked beam and considered the real-life test data which will contain the noise. In their paper the stochastic dynamic analysis of a cracked cantilever beam under white noise is addresses in the time domain.

Prashant Pawar and Ranjan Ganguli (ref. 54) proposed a genetic fuzzy system for damage detection in beams and helicopter rotor blades. They added noise in data to simulate the uncertainty present in experimental measurements and the modeling process. Given a computed frequency measurement delta $\Delta\omega$, random number u in the interval [–1, 1] and a noise lever parameter α, the noisy simulated data is given as $\Delta\omega^{noisy} = \Delta\omega^{fem} + u\alpha$. The parameter α defines maximum variance between the computed value of $\Delta\omega$ and simulated measured value $\Delta\omega^{noisy}$ which is a simulation of practical measurement. $\Delta\omega^{fem}$ is measurement delta obtained by finite element analysis. They conclude that the genetic fuzzy system gives an average success rate in damage detection of 99.81 percent, even with the noise levels of 0.20 in the data. For noise levels of 0.10 and 0.15, the average success rate is about 100 percent. The genetic fuzzy system is therefore very robust in the presence of significant noise inherence in mechanical system measurements.

Among ecologists, there has been a growing recognition of the importance of long-term correlations in environmental time series. John Halley (ref. 55) described that the family of 1/*f* noise-fluctuations defined in terms of the different timescales present- is a useful approach to this problem. White and random walk, the two currently favored descriptions of environmental fluctuations, lie at extreme ends of this family of process. Ecologists expect both rare and common events to be important. 1/*f* noise is a way describing these kinds of events. It is associated with an altogether slower decline in correlation. The extended family of 1/*f* noise is characterized by power-law spectra of forms: Spectral density, $S(f) \propto \frac{1}{f^{\gamma}}$, where $0 \le \gamma \le 2$. $S(f)$ is a decomposition of the noise signal into various component frequencies per unit frequency. By analogy with light, white noise is noise, whose spectral density is flat, containing equal amounts of all frequencies. The 1/*f* family takes its name from that member for which $\gamma \approx 1$, which is often called ‘pink noise’. It shows no preference for short or long timescale disturbances. He suggested that pink 1/*f* noise, which lies midway between white noise and random walk, might be the best null model of environmental variation.

1/*f* noise is widely present in electrical components. The presence of the excess low-frequency noise is particularly troublesome for microelectronics reliability, because its amplitude is much larger than that of other limiting noises. Keiji Takagi (ref. 56) provided experimental evidence of low-frequency noise in granular resistors. He measured the noise level with a spectrum or an FFT analyzer and a noise intensity

measuring system and noticed that all noise spectra are of the 1/*f* type in the low-frequency range. He explained a model of 1/*f* noise spectrum generation in granular structures. So there can be 1/*f* noise in the experiments using electronic devices.

Widespread occurrence of signals exhibiting power spectral density with 1/*f* noise behavior suggests that a general mathematical explanation of such an effect might exits. Mathematical algorithms and models for the generation of the processes with 1/*f* noise cannot, as a rule, be solved analytically and they do not reveal the origin as well as the necessary and sufficient conditions for the appearance of 1/*f* type fluctuations. Kaulakys and Meskauskas (ref. 57) proposed a simple analytical solvable model of 1/*f* noise. The models reveal main features, parameter dependencies and possible origin of 1/*f* noise, i.e., random increments of the time intervals between the pulses. The conclusion that $1/f^{\delta}$ noise with $\delta \approx 1$ may result from clustering of the signal pulses, particles or elementary events can be drawn from the analysis of the simple, exactly solvable models. The mechanism of clustering depends on the system.

Rangarajan (ref. 58) explained about random noise, structured noise, and physiological interference. Random noise: it refers to an interference that arises from a random process such as thermal noise in electrical devices. A random process is characterized by the probability density function (PDF) representing the probabilities of occurrence of all possible values of a random variable. It is common to assume the mean of random noise process to be zero. In most cases the noise is additive, $y(t) = x(t) + n(t)$. There are cases where the noise can be multiplicative, $y(t) = x(t) * n(t)$ where $y(t)$ is measures signal, $x(t)$ is original signal, and $n(t)$ is the noise. Structured noise: Power-line interference at 50 or 60Hz is an example of structured noise: the typical waveform of the interference is known in advance. It should, however, be noted that the phase of the interfering waveform will not usually be known. Physiological interference: The human body is a complex conglomeration of several systems and processes. Several physiological processes could be active at a given instant of time, each one producing many signals of different types. Physiological interference may not be characterized by any special waveform or spectra content, and is typically dynamic and non-stationary.

A fast growing variety of computer applications require use of random numbers. An immediate example can be found in a source of noise superimposed on the signals in studies of digital signal processing. Such random numbers are generated by iterative calculations based on a numerical formula. The sequences, generated based on the multiplicative congruential algorithm and its modifications that are often used in the functions generating random numbers for PCs are characterized periodically. These random numbers are not completely random. In view of the above mentioned facts, Shunsuke Kishimoto and Masuo Fukue (ref. 59) decided to develop so called "physical random digits," that is random digits generated by hardware and intended for safe use in PCs. The idea proposed by them consists in placement of all the hardware necessary for generation of physical random digits on a single IC chip. For this purpose, they used a Zener diode (semiconductor diode) that can be incorporated in the IC chip as a source of noise, and attempted to generate sequences of random digits based on the randomness of the short noise generated by the diode within time intervals of a fixed duration. This new method was verified by generating 1.2×10^9 decimal random integers and by testing such characteristics of the generated random number sequences as their uniformity and independence by statistical methods.

In control experiment, degraded images are obtained by adding noise to the original image and then they are restored by the restoration technique. In most cases, it is assumed that the noise is uncorrelated to the original image. Shan Suthaharan and Ray (ref. 60) introduced a method to generate a noise (called the first noise) such that average cross power spectrum between the first noise and the original image is zero. This method is extended to generate the second noise such that the average cross power spectrum between the second noise and the original image and the average cross power spectrum between the first and second noise are zero. Let *f* represent the original image. Suppose we generated, arbitrarily, an initial noise *w*1. Because of the very nature of its being generated arbitrarily, *w*1 could be correlated with *f*. To make the noise uncorrelated with *f*, they define first noise *n*1 as $n1 = w1 - (Cov(w1, f)/Var(f))*f$. Suppose a second initial noise *w*2 is generated arbitrarily. Because of the very nature of its being

generated arbitrarily, $w2$ could be correlated with f and $n1$. To make the noise uncorrelated they define second noise $n2$ as $n2 = w2 – (Cov\ (w2, n1)/Var(n1))*n1 – (Cov\ (w2, f)/Var(f))*f$.

A single set of noise-polluted data can lead to a biased result. The effect of the noise on the topology of the objective function is evident in the sensitivity of parameter estimates when noisy data are subjected to random perturbation. Pothisiri and Hjelmstad (ref. 61) adopted a data perturbation scheme to create a sample of artificial data sets for parameter estimation at each stage of the damage localization process. They explained the solution multiplicity for the noisy data. To wit, the j^{th} component of perturbed eigenvector from the j^{th} component of the i^{th} measured eigen vector as $\tilde{\phi}_{ij} = \hat{\phi}_{ij}(1 + \eta_{ij})$, where η_{ij} is a uniform random variate in the range $[–a, a]$. The amplitude a must be specified to account for the sensitivity of parameter estimate due to noise. Once the multiple solutions have been located using the random starting points, the data perturbation scheme can be applied to obtain the sensitivity information for each of the solutions by using the individual solution as a fixed starting point for each perturbed data set.

2.6 Literature Review—Application of Wavelet Theory for Denoising and Signal Feature Reconstruction

In all the experiments, presence of noise is a major issue. Denoising of the signals and images has taken major role in the current research activities. In the last two decades, vast research is going on wavelet theory and their applications. One of the applications of the wavelets is denoising of the signals and images. The wavelet transform has proven to be very successful in making signal and noise components of the signal distinct. The signal is transformed into some domain where the noise component is more easily identified, thresholding operation is then applied to remove the noise, and finally the transformation is inverted to reconstruct a (hopefully) noise-free signal.

In the recent years there has been a fair amount of research on wavelet thresholding and threshold selection for signal de-noising, because wavelet provides an appropriate basis for separating noisy signal from the signal. Lakhwinder Kaur, Savita Gupta, and Chauhan (ref. 62) proposed a near optimal threshold estimation technique for image denoising which is subband dependent i.e., the parameter for computing the threshold are estimated from the observed data, one set for each subband. The proposed method is called “NormalShrink”. The threshold value (T_N) is computed by $T_N = \beta\ \sigma^2 / \sigma_y$. β is scale parameter, σ^2 is variance and σ_y is standard deviation of the subband. The image denoising algorithm is described and experiments are conducted to assess the performance of NormalShrink. The results show that it removes noise significantly.

Under the framework of VC-theory, wavelet thresholding amounts to ordering of wavelet coefficients according to their relevance to accurate function estimation, followed by discarding insignificant coefficients. Existing wavelet thresholding methods specify an ordering based on the coefficient magnitude, and use threshold(s) derived under Gaussian noise assumption and asymptotic settings. In contrast, Shi Zhong and Vladimir Cherkassky (ref. 63) proposed a wavelet threshold approach, uses orderings based on the coefficient magnitude, which better reflects statistical properties of natural images, and VC-based thresholding developed for finite sample settings under very general noise assumptions. They interpreted image denoising as a special case of signal estimation problem and propose a model selection based denoising method under the framework of VC theory, which was developed for estimating data dependencies from the finite samples. VC-theory provides a framework for model selection called structural risk minimization (SRM). Under SRM, a set of possible models ordered according to their complexity. Level dependent thresholding has been proposed to improve the performance of wavelet thresholding method. Instead of using a global threshold, level-dependent thresholding uses a group of thresholds, one for each scale level.

Microarray imaging is a recent cutting-edge technology in bioinformatics which can monitor thousands of genes simultaneously. This technology can monitor thousands of DNA sequences in a high density array on a glass. The noise source in microarray imaging originates from different sources during the course of experiment, such as photon noise, electron noise, laser light reflection, dust on the slide, and so on. Hence, it is crucial to denoise the resultant image within this process. Wang, Robert Istepanian, and Yong Hua Song (ref. 64) proposed a new approach on wavelet theory to provide an enhanced approach for eliminating such noise source and ensure better gene expression. They apply the SWT (Stationary Wavelet Theory) method to preprocess the microarray images for removing the random noise. In this method at each level, when the high-pass and low-pass filters are applied to the data, the two new sequences have the same length as the original sequences. To do this, the original data is not decimated. However, the filters at each level are modified by padding them out with zeros. They apply the biorthogonal wavelet procedure to decompose the image using Matlab. SureShrink thresholding algorithm is applied to the subimages. The results show that SWT provides a better performance over traditional wavelet transform method.

Ultrasonic is the perfect means for morphological investigation of the neonatal brain. A problem, common to all medical ultrasound images, is the presence of speckle noise, which not only complicates the visual interpretation of images, but also the quantitative measurements. Suppression of speckle in noise is necessary to get reliable measurements. Ivana Duskunovic, et al.(ref. 65) compared two noise removal techniques, based on wavelet decomposition, applied to speckle images. The first method applies Bayesian shrinkage of the wavelet coefficients using MRF model to express the prior knowledge about the spatial clustering of the coefficients. This method works as follows: Take the discrete wavelet transform of the observations, apply a simple nonlinearity (shrink towards zero) to each wavelet coefficient, and compute signal estimates by applying the inverse transform to the transformed coefficients. The second method proposed is based on the spatial analysis of detail images and determining the position of real edges and removing false edges. Values of the pixels, for which they determine that do not belong to an edge, are set to zero. Specifically, their method operates on detail images. Each technique has its advantages. The first technique provides significant noise suppression, which results in very smooth areas uniformly corrupted by noise. The second technique does not suppress as well as the first but preserves sharpness better. The techniques proposed yield comparable results.

Gaussian smoothing is a simple denoising method and it is widely used in computer vision algorithms. However, Gaussian smoothing destroys the structure of image data. The orthonormal basis property of wavelet brings the possibility of denoising without destroying the structure of input data. Qi Li, et al. (ref. 66) investigated how useful the wavelet denoising technique is in improving the accuracy of computing fundamental matrices. They applied a methodology called waveShrink, which has wide applications and implemented in commercial software, e.g., wavelet toolbox of Matlab. Determining threshold is key issue in waveShrink denoising. They applied minimax threshold technique to denoise the input data. The experimental results show that wavelet denoising can help improving accuracy of fundamental matrix on a large scale of images while it is useless for stereo images containing randomized information or highly regular texture information.

2.7 Concluding remarks

A great deal of research in the past thirty years has been aimed at establishing an effective method for health monitoring in civil, mechanical, and aerospace structures. The objective is to determine the existence, location, and degree of damage in a structure. The development of a successful technology for structural health monitoring has enormous potential for application in evaluation of offshore structures and bridges subject to fatigue, corrosion, impact and earthquakes as well as buildings and aerospace structures subject to severe loads or structural deterioration. A variety of methods for evaluating damage in structural systems have emerged and evolved. All of these methods require a parameter estimation algorithm to drive them; i.e., the selection of an appropriate “measure” or suitable perturbation in system

properties. To this end, recent work by Saleeb, et al. (ref. 67) has been directed towards the development of appropriate global indices of this type, based on a more fundamental approach in structural mechanics.

The novelty of this approach stems from the use of an alternative formulation in material (vs. physics) space. These dual balance laws are revealing in that the resulting force/source terms are directly and explicitly driven by increased heterogeneity due to deterioration. It thus provides an ideal candidate for the damage detection parameter. Such a damage parameter is easily computed from the measured raw data (e.g., strains, deflections, and rotations). A large number of numerical simulations and comparison to data have clearly demonstrated the power of this formulation under different test conditions (refs. 68 and 69).

Most of the experiments contain some amount of noise in the measured data. The damage identification formulation should be robust enough to sustain the noise to identify the damage. For the past few decades there has been decent amount of research in the field of development of the filters to reduce the noise in signal processing and image processing. Few filtering techniques are developed for special applications. The important feature of the filter is to retain the significant details. For the last two decade there is great deal of research is going on wavelet theory. One of the applications of the wavelets is denoising the data. It has the feature to retain the significant details. Plenty of wavelets are developed by the researchers, but all wavelets will not fit to particular data. Hence selecting the suitable wavelet for the study takes major role. The focus of this study is to provide further validation for the feasibility and effectiveness of this new algorithm under realistic conditions. Algorithm is evaluated with noisy data. Wavelet theory is adopted to reduce the intensity of the noise in the simulated data.

Chapter III
Theoretical Developments

The theoretical background necessary for this study is described in this chapter. This chapter is divided into two sections; Damage Identification Technique and Wavelet Theory. The damage identification technique that will be presented is utilized for the study of damage detection as applied to plate like structures and a unique application of wavelet theory is used to reduce the noise intensity of the measured data required in the detection scheme.

3.1 Damage Identification Technique

3.1.1 Introduction.—This section deals with the description of the underlying mathematical structure of the damage index used in the detection scheme and various methods to extract the damage sensitivity features necessary for visualization in the terms of pattern recognition. In this regard, it will be gathered from subsequent discussions that, despite its conceptual simplicity and theoretical tractability, the damage index possess a rather intricate vectorial/tensorial character. As a result, it is best to approach the pattern recognition strategy from the viewpoints of both intensity (magnitude) as well as directional properties. For instance, the magnitudes of various components of the damage tensorial index are shown to exhibit large and abrupt changes (spikes/peaks) in the presence of "true" defects. However, taken in isolation, this representation alone is not sufficient, since also vibration response changes due to environmental and/or operational variability (i.e., the so-called false alarm tests, which are known to pose extreme difficulties for many existing detection schemes) can trigger spurious patterns of this type.

Hence, the directional property of the damage index is quite unique. The vector flow fields evaluated form the projection of the tensor component on a (variation – consistent C° – field of) position vectors perturbations will always be pointing in the direction of increased dissipation. In other words, vectors perturbations will point in the direction of decreased total stored energy. Consequently, for true defect case (irrespective of the underlying physical mechanism leading to the defect/deterioration) the discrete set of arrows representing these vector fields on the boundaries of the regions enclosing the defects will be directed outwards. This will persist in a consistent manner, irrespective of the mechanical response signature being interrogated, i.e., any vibration mode or any static load testing. On the contrary, this position will be completely or partially lost in case of false alarm test, e.g., all vectors will not be reversed, for extreme case of increase stiffness in a small localized region in the structure, to point inwardly.

3.1.2 Governing Equations for Flexural Vibration of Shear-Flexible Plate.—Consider the case of mechanical response of a homogenous, isotropic plate (with no thermal effects, etc.) with no defects subjected to free flexural vibrations. Adopting the well-known approach of treating free vibration, we consider for one typical mode; i.e., the n^{th} mode with frequency $\omega \equiv \omega_n$ and mode shapes $(w,\ \psi_1,\ \psi_2)$ $\equiv (w_n,\ \psi_{1n}\ ,\psi_{2n})$. Here w, ψ_1, ψ_2 represents the displacement in *z*-direction and the rotations in the *x* and y- directions respectively. Note that, for convenience, we will drop the subscript '*n*' in all the subsequent derivations. A shear – flexible theory developed by Mindlin/Reissener(ref. 77) is utilized for plate flexure. This shear – flexible theory has both bending and transverse shear effects as well as lateral (linear) and rotary inertia accounted for. In addition, the following notation is introduced.

The kinetic quantities, namely, the curvatures and the transverse shear may be expressed as,

- Curvatures:

$$\kappa_x = \frac{\partial^2 \psi_1}{\partial x^2}, \qquad \kappa_y = \frac{\partial^2 \psi_2}{\partial y^2}, \qquad \kappa_{xy} = \frac{\partial \psi_1}{\partial x} + \frac{\partial \psi_2}{\partial y}$$

- Transverse Shears:

$$\tau_{xy} = \frac{\partial w}{\partial x} + \psi_1 , \qquad \tau_{yz} = \frac{\partial w}{\partial y} + \psi_2$$

The static quantities, i.e., the shear resultants per unit width which taken from the moments are expressed as,

- Moments:

$$M_x \equiv M_{11} = \int_{-h/2}^{h/2} \sigma_{xx} z dz$$

$$M_y \equiv M_{22} = \int_{-h/2}^{h/2} \sigma_{yy} z dz$$

$$M_{xy} \equiv M_{12} \equiv M_{21} = \int_{-h/2}^{h/2} \sigma_{xy} z dz$$

The constitutive relationships for the plate isotropy and with the assumptions of linear-elastic response take the following form,

$$M_{11} = D_b \left(\frac{d\psi_1}{dx} + v \frac{d\psi_2}{dy} \right)$$

$$M_{22} = D_b \left(\frac{d\psi_2}{dy} + v \frac{d\psi_1}{dx} \right)$$

$$M_{12} = M_{21} = D_b \left(\frac{1-v}{2} \right) \left(\frac{d\psi_1}{dy} + \frac{d\psi_2}{dx} \right)$$

$$Q_1 = k^2 Gh \left(\frac{d\psi_1}{dx} + \psi_1 \right)$$

$$Q_2 = k^2 Gh \left(\frac{d\psi_2}{dy} + \psi_2 \right)$$

Where

$$D_b = \left(\frac{Eh^3}{12(1-v^2)} \right)$$

Finally in what follows are the well-known conservation expressions in physical space. That is, the balance of the linear and angular momentum in the form of the first order differential equations is given as follows:

$$\frac{dM_{11}}{dx} + \frac{dM_{12}}{dy} - Q_1 + \rho I \omega^2 \psi_1 \equiv 0 \tag{3.1.1}$$

$$\frac{dM_{12}}{dx} + \frac{dM_{22}}{dy} - Q_2 + \rho I \omega^2 \psi_2 \equiv 0 \tag{3.1.2}$$

$$\frac{dQ_1}{dx} + \frac{dQ_2}{dy} + \rho h \omega^2 w \equiv 0 \tag{3.1.3}$$

The following sections present a step-by-step procedure from which we can show that the equations (3.1.1), (3.1.2), and (3.1.3) are the basic equations for the Defect Energy Parameter.

3.1.3 The Detection Parameter Specialization.—The total strain energy, W, for thin plate can be defined as

$$W(w,\psi_1,\psi_2) = \frac{1}{2}\left[M_{11}\frac{\partial \psi_1}{\partial x} + M_{22}\frac{\partial \psi_2}{\partial y} + M_{12}\left(\frac{\partial \psi_1}{\partial y} + \frac{\partial \psi_2}{\partial x}\right) + Q_1\left(\frac{\partial w}{\partial x} + \psi_1\right) + Q_2\left(\frac{\partial w}{\partial y} + \psi_2\right)\right] \tag{3.1.4}$$

After substituting the expression for M_{11}, M_{22}, M_{12}, Q_1 and Q_2 in equation (3.1.4) the following expression for W is obtained

$$\begin{aligned} W(w,\psi_1,\psi_2) = {} & \frac{1}{2}Ghk^2\psi_1^2 + \frac{1}{2}Ghk^2\psi_2^2 + Ghk^2\psi_2^2\left(\frac{\partial w}{\partial y}\right)^2 + \frac{1}{2}Ghk^2\left(\frac{\partial w}{\partial y}\right)^2 + \frac{1}{4}D_b\left(\frac{\partial \psi_1}{\partial y}\right)^2 \\ & - \frac{1}{4}D_b\nu\left(\frac{\partial \psi_1}{\partial y}\right)^2 + \frac{1}{2}D_b\left(\frac{\partial \psi_2}{\partial y}\right)^2 + Ghk^2\psi_1^2\left(\frac{\partial w}{\partial x}\right) + \frac{1}{2}Ghk^2\left(\frac{\partial w}{\partial x}\right)^2 \\ & + D_b\nu\left(\frac{\partial \psi_2}{\partial y}\right)\left(\frac{\partial \psi_1}{\partial x}\right) + \frac{1}{2}D_b\left(\frac{\partial \psi_1}{\partial x}\right)^2 + \frac{1}{2}D_b\left(\frac{\partial \psi_1}{\partial y}\right)\left(\frac{\partial \psi_2}{\partial x}\right) \\ & - \frac{1}{2}D_b\nu\left(\frac{\partial \psi_1}{\partial y}\right)\left(\frac{\partial \psi_2}{\partial x}\right) + \frac{1}{4}D_b\left(\frac{\partial \psi_2}{\partial x}\right)^2 - \frac{1}{4}D_b\nu\left(\frac{\partial \psi_2}{\partial x}\right)^2 \end{aligned}$$

$$\text{The kinetic energy per unit area, } T = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{1}{2}\rho V^T V dz \tag{3.1.5}$$

where the velocity vector, $V = \left[x_3\frac{d\psi_2}{dt}, x_3\frac{d\psi_1}{dt}, \frac{dw}{dt}\right]$

After integrating over the thickness of the plate, T reduces to (for the special case of free vibrations; i.e., over one natural period of vibration in the typical mode number "n")

$$T = \frac{1}{2}\rho\omega_n\left[\frac{1}{12}h^3\left(\psi_1^2 + \psi_2^2\right) + hw^2\right] \tag{3.1.6}$$

The first term in equation (3.1.6) represents the rotary inertia due to the rotations ψ_1 and ψ_2 along the *x* and *y*-axes, and the second term is the translation/transverse inertia due to the displacement *w* along the *z*-axis.

The defect energy parameter for detection can then be formed as two components of "forces", i.e.,

$$\left.\begin{aligned} F_x \equiv F_1 &= \int_s \left(P_{11} l + P_{21} m\right) ds \\ F_y \equiv F_2 &= \int_s \left(P_{12} l + P_{22} m\right) ds \end{aligned}\right\} \tag{3.1.7}$$

Where *l* and *m* are the direction cosines of the unit normal to "*ds*", and *s* is the length of the perimeter curve surrounding an area of the plate middle surface, the terms P_{11}, P_{12}, $P_{21,\text{ and}}\, P_{22}$ are given as,

$$\left.\begin{aligned} P_{11} &= (W - T) - \left(M_{11} \frac{d\psi_1}{dx} + M_{12} \frac{d\psi_2}{dx} + Q_1 \frac{dw}{dx} \right) \\ P_{21} &= -\left(M_{12} \frac{d\psi_1}{dx} + M_{22} \frac{d\psi_2}{dx} + Q_2 \frac{dw}{dx} \right) \\ P_{12} &= -\left(M_{11} \frac{d\psi_1}{dy} + M_{12} \frac{d\psi_2}{dy} + Q_1 \frac{dw}{dy} \right) \\ P_{22} &= (W - T) - \left(M_{12} \frac{d\psi_1}{dy} + M_{22} \frac{d\psi_2}{dy} + Q_1 \frac{dw}{dy} \right) \end{aligned}\right\} \tag{3.1.8}$$

In the absence of any "defects" in the plate structure, the associated forces in eq. (3.1.7) must be zero and equivalently this will lead to the following condition:

$$\text{Divergence of tensor } P = 0. \tag{3.1.9}$$

The above "proof" will serve to facilitate the understanding of the abstract, "mathematical-intriguing" nature of the damage parameter components. Note that, the formal derivation of the proper expression for such damage parameters requires the study of the variational symmetry of complex structures governed by coupled systems of partial differential equations. Using the language of "finite-strain" analysis, one then considers the inverse motion of the structure i.e., variations in material space while then present deformed state remain unchanged.

According to the defect energy point of view, the basic parameters in the equations (3.1.7) to (3.1.9) convey the information about the damage complete and clear. One can consider changes in response due to the defect energy equations to give changes in natural frequency or mode shapes by making some approximations and assumptions. Though this is useful in some cases, the ensuring detection indices will certainly be limited in their scope for general applications. However, if there are no approximations, all individual contributions such as natural frequency, deformations, stresses, strains, shear and bending forces, etc. interact in a very complex manner with their respective different weights and degrees of participation on a particular excitation to produce the final damage index. Here, the latter argument is adopted in more comprehensively.

3.1.4 Discussion and Generalization.—With above in mind, one can obtain the necessary expressions required for the analysis of other structural vibration modes; e.g., the membrane/inplane counterpart. In fact, a more effective approach is to consider the combined flexure/membrane formulation

as in the case of general, spatially curved shells. This indeed was the case considered in the implementation completed for the purpose of the feasibility study for the damage detection scheme.

In particular, the details of the formulation utilized here as a basis for the numerical simulations is given in (ref. 80); i.e., for general mixed-type, shell finite elements. In the sequel, only a very brief outline of the schemes will be given. As to the simulation of damage scenarios, the simplest types of defects have been considered; i.e., in the form of "elastic" material stiffness degradation. Note that this approach has also been the most popular approach in the existing literature on damage detection. Further generalization to more complex types of dissipative damage/defect phenomena should be straightforward, both conceptually as well as mathematically.

3.1.5 Background on Simulations.—Theoretical development of the damage index is based on the shear flexible theory of Mindlin/Reissner (ref. 77). This theory includes rotary inertia and shear in the same manner as Timoshenko's (ref. 81) one-dimensional theory of bars.

Depending upon the treatment of the effect of transverse shear deformation, the existing plate bending elements may be divided into two groups: one based on Kirchhoff Plate Theory and the other on Mindlin Plate Theory. In the formulation according to Kirchhoff, finite elements derived from the principle of virtual work or the principle of minimum total potential energy must satisfy the C^1 compatibility across element boundaries. On the other hand, the Mindlin plate formulation, that includes the effect of transverse shear deformation, C^1 compatibility is not required, even an element derived from the principle of minimum total potential energy. In addition, a Mindlin plate bending element can be easily extended to a degenerate type shell element, with curved geometry. But unfortunately Mindlin plate bending elements have a tendency to lock as the thickness of the plate becomes very small. Therefore, in formulating a Mindlin plate bending element, special care must be taken to eliminate locking. A similar locking effect has been discussed by Hinton, et al. (ref. 82).

The development of suitable finite element models for linear and nonlinear analysis of plates and shells has always presented a major challenge, due to the many theoretical intricacies involved. To overcome such intricacies, Saleeb, et al.(ref. 79) developed a simple, shear flexible, quadrilateral plate element, designated as HMPL5. To predict the capabilities of the damage index, this quadrilateral plate element has been used for the present computer simulations. This element has five nodes, with three displacements and two rotations at each node. The interior fifth node is located at the geometrical centroid of the element. As Mindlin plate theory accounts for transverse shear in addition to bending deformation, this shear flexible quadrilateral plate element is valid for thin as well as thick plates.

3.1.6 Outlines for Shell Modeling in Numerical Simulations.—Based on a modified Hellinger-Reissner variational principle, where both displacement and strain fields are assumed independent, Saleeb, et al.(refs. 78, 79, and 80) developed an effective 5-node shell element, designated as HMSH5. The HMSH5 element is primarily used for damage numerical simulations. Five degrees of freedom are defined at each nodal point, that is, three translations (*u, v, w*) along the Cartesian global axes and two rotations (θ_1, θ_2) about mutually perpendicular lamina coordinates. In total the HMSH5 element has 25 DOF. In finite element descritizations using element type HMSH5, the displacements are interpolated in terms of nodal degrees of freedom and can be written as

$$u = N\,q \tag{3.1.10}$$

where N is the interpolation matrix and q is the vector of nodal displacements of the element where

$$q = \left[u_1, v_1, w_1, \theta_1^{(1)}, \theta_2^{(1)}, \ldots\ldots\ldots\ldots, u_5, v_5, w_5, \theta_1^{(5)}, \theta_2^{(5)} \right]^T \tag{3.1.11}$$

in which the superscript T denotes transpose of a matrix. Consequently, the acceleration field, $\ddot{u}$, within the element is interpolated in terms of nodal acceleration as

$$\ddot{u} = N\,\ddot{q} \tag{3.1.12}$$

while strain ε are approximated in terms of a strain parameter β, as

$$\varepsilon = p\beta \tag{3.1.13}$$

where β has 19 terms, among which seven terms that belong to the 'membrane' lamina strains (constant through thickness) and 12 terms are included in the combined bending part. In the combined bending portion, five terms are the transverse shear strain components. *P* is a (5×19) strain interpolation matrix for element lamina strains. In general, the entries in *P* are polynomial functions of natural coordinates. By utilizing equations (3.1.10) and (3.1.13), the Hellinger-Reissner functional π_R can be written as

$$\pi_R = q^T M\,\ddot{q} - \frac{1}{2}\beta^T H\beta + \beta^T G q - W_{ext} \tag{3.1.14}$$

Where

$$H = \int_v P^T D P dv\,, \qquad G = \int_v P^T D B^1 dv\,, \quad \text{and} \qquad M = \int_v N^T \rho N dv$$

In the above, *D* is the elastic stiffness matrix, B^1 is local strain displacement matrix and W_{ext} is the external work done. Invoking the stationary behavior of equation (3.1.14) with respect to strain yields β in terms of *q*

$$\beta = H^{-1} G q \tag{3.1.15}$$

which is used to eliminate the strain parameters on the element level. Finally substituting equation (3.1.15) into (3.1.14) results in

$$\pi_R = q^T M\,\ddot{q} + \frac{1}{2} q^T K q - W_{ext} \tag{3.1.16}$$

where the stiffness matrix for the hybrid/mixed element is given by

$$K = G^T H^{-1} G \tag{3.1.17}$$

Once the stiffness and mass matrices are assembled for the entire shell, the equilibrium equation can be obtained as

$$M\,\ddot{q} + Kq = 0 \tag{3.1.18}$$

which leads to the standard generalized eigen-problem to be solved for ω and the corresponding eigen-vector, ϕ; e.g, subspace iteration (Bathe [ ])

$$\left(K - \omega^2 M\right)\phi = 0 \tag{3.1.19}$$

3.1.7 Damage Scenario.—When damage develops, effective properties of the material in the damaged area of the structure are changed. These changes in the material properties may be characterized by modification of the stiffness as well as the damping coefficient which pertains to the vibration behavior. In the present study, a 2-D plate was chosen for damage detection using the defect energy parameter since detailed investigations of other beam and frame like structures has been already conducted.

For each plate, several damage locations are imposed. These can be a single or multiple situations that may be imposed at any span of the plate, interior or exterior, close to or away from prescribed boundary conditions. For each location, different amounts of damage severity may be imposed. Damage severity is represented by the percentage reduction of bending rigidity, EI, as well as shear rigidity GA. In this study, the percentage of damage is defined as that specific percentage reduction of Young's Modulus (E) at the specified element. For example, "1 percent damage at element 120" means a 1 percent reduction of Young's Modulus (E), i.e., 0.99E is used in element 120 in the finite element model. Other material properties such as cross-section, mass and moment of inertia remain intact. Along with the percentage reduction in the Young's Modulus, the damage also simulated as a line crack by creating multiple nodes at the same geometric location and changing the nodal connectivity, and also by eroding an element to create a tiny hole in the structure.

3.2 Wavelet Theory

3.2.1 Introduction.—The concept of wavelets has been discussed in the literature since the early 1800's. However, only in recent years has significant progress been made in the applications of the wavelets to signal processing. Historically mathematics, physics, electrical engineering, and other applied sciences have contributed to the development of the wavelet theory. For example, significant practical applications of wavelets have been found in signal and image processing, wireless communications, and control system analysis, which are in the electrical engineering domain. The pioneering work of Daubechies in the early 1980s has shown effective use of wavelets in so-called interdisciplinary research. Other fields that are making use of wavelets include astronomy, acoustics, nuclear engineering, signal and image processing, music, magnetic resonance imaging, speech discrimination, optics, earthquake-prediction, radar, human vision, and pure mathematics applications such as solving partial differential equations.

If we consider signal processing, wavelets are used for transforming a signal into a form that it is more useful to the application. For example, if we are interested in reducing the noise in a signal, the best representation is the one in which the signal and noise are easily separated. The two most common representations for a one-dimensional signal are the time signal, and its Fourier transform. Unfortunately, it is not easy to extract frequency information from the time signal and vice versa. Different signal representations depend on the choices of the time-frequency resolution. The choice of the proper signal processing technique is based on the signal that needs to be analyzed. Short-time Fourier transform (STFT) and the continuous wavelet transform (CWT) are used to represent a signal in both the time and frequency domains.

3.2.2 Continuous Wavelet Transforms.—The wavelet transform maps a function, s(t), into a two-dimensional domain and is denoted by

$$W_s(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} s(t) h^* \left(\frac{t-b}{a} \right) dt \qquad (3.2.1)$$

$$= \int_{-\infty}^{+\infty} s(t) h_{ab}^* (t)\, dt$$

where $h(t)$ is in general called the mother wavelet, and the daughter wavelets, are given by:

$$h_{ab}(t) = \frac{1}{\sqrt{a}} h\left(\frac{t-b}{a}\right) \tag{3.2.2}$$

Equation (3.2.1) is also known as the forward transform or analysis; $h_{ab}(t)$ is a set of basis functions/daughter wavelets obtained from the mother wavelet $h(t)$ by compression or dilation using the scaling parameter a and translation using the shift parameter b. The scaling parameter a is positive and varies from 0 to ∞. For $a < 1$, the transform performs compression of the signal, and for $a > 1$, the transform performs dilation of the signal. The reason for choosing the factor $1/\sqrt{a}$ in the above equation is to keep the energy of the daughter wavelets constant. Without this normalization factor, for different a values the wavelets dilate or compress and their total energy changes.

The inverse wavelet transform is given by:

$$s(t) = \frac{1}{c} \int_{-\infty}^{+\infty} \int_{0}^{\infty} W_s(a,b) h\left(\frac{t-b}{a}\right) \frac{da}{a^2} db \tag{3.2.3}$$

in which the constant c is defined as

$$c = \int_{-\infty}^{\infty} \frac{|H(\omega)|^2}{\omega} d\omega < \infty \tag{3.2.4}$$

Equation (3.2.3) is also referred to as the reconstruction formula, inverse transform, or synthesis, and equation (3.2.4) is known as the admissible condition.

The wavelet theory as applied in the present research, is utilized for the reconstruction of the damage "signal" from the noisy data. In this context, it is necessary to treat the damage signal as a discrete signal. Therefore, discrete wavelet transforms are used to complete the study. The following section describes the theory behind discrete wavelet transforms in detail.

3.2.3 Discrete Wavelet Transforms.—A discrete signal is a function of time with values occurring at discrete instants. Generally we shall express a discrete signal in the form $f = (f_1, f_2, f_3, ..., f_N)$, where N is a positive even integer which we shall refer to as the length of f. The values of f are generated by sampling a continuous signal g at uniformly spaced time intervals, i.e., $t = t_1, t_2, ..., t_N$. The values of f are

$$f_1 = g(t_1), f_2 = g(t_2), ..., f_N = g(t_N) \tag{3.2.5}$$

All wavelet transforms decompose a discrete signal into two subsignals. Each of these subsignals is half the length of the original signal. One subsignal is referred to as the average or *trend* while other subsignal is referred to as the difference or *fluctuation*. The first trend subsignal, $a^1 = (a_1, a_2, ..., a_{N/2})$, for the signal f is computed by taking the average of the values in f. The first fluctuation of the signal f, which is denoted by $d^1 = (d_1, d_2, ..., d_{N/2})$, is computed by taking difference of the values in f.

To further clarify the concept of the discrete wavelet transform (DWT), the Haar wavelet, which is the simplest type of wavelet, is taken as an example and will provide a good foundation to understand the more sophisticated wavelet transforms.

3.2.3.1 Haar Wavelet Transform: The Haar transform of the signal $f = (f_1, f_2, f_3, ..., f_N)$ will decompose the signal into averages, *a*, and differences, *d*. The decomposition can be done at multiple levels. The first level is the mapping H_1, defined by

$$f \xrightarrow{H_1} (a^1 \mid d^1) \tag{3.2.6}$$

Averages: $a^1 = (a_1, a_2, ..., a_{N/2})$

$$\left.\begin{aligned} a_1 &= \frac{(f_1 + f_2)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_1 + f_2)}{2}\right) \\ a_2 &= \frac{(f_3 + f_4)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_3 + f_4)}{2}\right) \\ a_{N/2} &= \frac{(f_{N-1} + f_N)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_{N-1} + f_N)}{2}\right) \end{aligned}\right\} \tag{3.2.7}$$

Fluctuations: $d^1 = (d_1, d_2, ..., d_{N/2})$

$$\left.\begin{aligned} d_1 &= \frac{(f_1 - f_2)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_1 - f_2)}{2}\right) \\ d_2 &= \frac{(f_3 - f_4)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_3 - f_4)}{2}\right) \\ d_{N/2} &= \frac{(f_{N-1} - f_N)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_{N-1} - f_N)}{2}\right) \end{aligned}\right\} \tag{3.2.8}$$

Inverse Haar transform from level-1 decomposition:

$$(a^1 \mid d^1) = \left(\frac{a_1 + d_1}{\sqrt{2}}, \frac{a_1 - d_1}{\sqrt{2}}, \frac{a_2 + d_2}{\sqrt{2}}, \frac{a_2 - d_2}{\sqrt{2}}, ..., \frac{a_{N/2} + d_{N/2}}{\sqrt{2}}, \frac{a_{N/2} - d_{N/2}}{\sqrt{2}}\right) = f \tag{3.2.9}$$

The second level Haar transform decomposes the trend subsignal a^1. The second level mapping H_2, defined by

$$f \xrightarrow{H_2} (a^2 \mid d^2 \mid d^1) \tag{3.2.10}$$

And the *N* level mapping H_N, defined by

$$f \xrightarrow{H_N} (a^N \mid d^N \mid ..... \mid d^2 \mid d^1) \tag{3.2.11}$$

3.2.3.2 Haar wavelets and scaling signals: In this section we further discuss the Haar wavelets and the scaling signals, and defining the averages and fluctuations as an inner product of these signals. The 1-level Haar wavelets are defined as

$$\left.\begin{aligned} W_1^1 &= \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, 0, ..., 0\right) \\ W_2^1 &= \left(0, 0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, 0, ..., 0\right) \\ &. \\ &. \\ W_{N/2}^1 &= \left(0, 0, ..., 0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right) \end{aligned}\right\} \tag{3.2.12}$$

Consider the inner product of f and W_1^1,

$$\left.\begin{aligned} \left\langle f, W_1^1 \right\rangle &= \left(f_1 \frac{1}{\sqrt{2}} + (-f_2 \frac{1}{\sqrt{2}}) + 0 + ... + 0\right) \\ &= \frac{f - f_2}{\sqrt{2}} = d_1^1 . \end{aligned}\right\} \tag{3.2.13}$$

Therefore, we can summarize that, 1-level differences are defined as,

$$d_m^1 = \left\langle f, W_m^1 \right\rangle \tag{3.2.14}$$

in which the 1-level scaling signals are defined as

$$\left.\begin{aligned} V_1^1 &= \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, ..., 0\right) \\ V_2^1 &= \left(0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, ..., 0\right) \\ &. \\ &. \\ V_{N/2}^1 &= \left(0, 0, ..., 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \end{aligned}\right\} \tag{3.2.15}$$

In addition, consider the inner product of f and V_1^1 as,

$$\left.\begin{aligned} \left\langle f, V_1^1 \right\rangle &= \left(f_1 \frac{1}{\sqrt{2}} + f_2 \frac{1}{\sqrt{2}} + 0 + ... + 0\right) \\ &= \frac{f + f_2}{\sqrt{2}} = a_1^1 . \end{aligned}\right\} \tag{3.2.16}$$

Therefore, we can summarize that, 1-level averages, are given as

$$a_m^1 = \left\langle f, V_m^1 \right\rangle \tag{3.2.17}$$

Thus in general,

$$\text{Averages, } a_m^N = \left\langle f, V_m^N \right\rangle \text{ and Differences, } d_m^N = \left\langle f, W_m^N \right\rangle \tag{3.2.18}$$

and the inverse Haar transform as an inner product is defined as,

$$f = A^k + D^k + \ldots\ldots + D^2 + D^1 \tag{3.2.19}$$

where, $A^k = \left\langle f, V_m^k \right\rangle V_m^k$ and $D^k = \left\langle f, W_m^k \right\rangle W_m^k$

Equation (3.2.19) shows how a discrete signal is synthesized by beginning with a low resolution signal and successively adding on "details" to create a high resolution signal, ending with a complete synthesis of the signal at the finest resolution. This process is known as "multiresolution analysis (MRA)".

The typical application of the Haar transforms is compression of a signal and removing noise from a signal. In this report, wavelets are used to reduce the noise intensity in a signal. Therefore, we will focus this particular application and the following sub-section describes on the process of noise removal using the Haar wavelet.

3.2.3.3. Denoising of a Signal Using Haar Transforms: A basic signal model gives the measured signal as actual signal plus its noise, i.e.,

$$f = s + n$$

where f = Measured signal
s = Actual signal
n = Noise

Note the term noise is referred to as the disturbance in the measured signal. To remove or to reduce the intensity of the noise in a signal, the following procedure is used:

(1) Take the wavelet transform of signal *f*.
(2) Choose a threshold T, so that the noisy signal's transform values have magnitudes which lie below *T*.
(3) Set values of the transform to zero which are less than the threshold *T*.
(4) Take the inverse transform to get the denoised signal.

In this research, Daub4 wavelet from Daubechies family is utilized. The Daubechies wavelet transforms are defined in the same way as the Haar transform by computing averages and differences via scalar products with scaling signals and wavelets. The only difference is how the scaling signals and wavelets are defined.

3.2.3.4 The Daub4 Wavelets and Scaling Signals: In this section, details of the Daub4 wavelet are presented. First, the *scaling* numbers $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are defined by

$$\alpha_1 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \alpha_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \alpha_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \alpha_4 = \frac{3-\sqrt{3}}{4\sqrt{2}} \tag{3.2.20}$$

Using these scaling numbers, the 1-level Daub4 scaling signals are expressed as,

$$\left.\begin{aligned} V_1^1 &= (\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, ..., 0) \\ V_2^1 &= (0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, ..., 0) \\ V_3^1 &= (0, 0, 0, 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, ..., 0) \\ &\vdots \\ V_{\frac{N}{2}-1}^1 &= (0, 0, ..., 0, \alpha_1, \alpha_2, \alpha_3, \alpha_4) \\ V_{\frac{N}{2}}^1 &= (\alpha_3, \alpha_4, 0, 0, ..., 0, \alpha_1, \alpha_2) \end{aligned}\right\} \quad (3.2.21)$$

Let us define,

$$\begin{aligned} V_1^0 &= (1, 0, ..., 0) \\ V_2^0 &= (0, 1, 0, ..., 0) \\ &. \\ &. \\ V_2^0 &= (0, 0, ..., 0, 1) \end{aligned} \quad (3.2.22)$$

Using the equation (3.2.22), the first level Daub4 scaling signals satisfy

$$V_m^1 = \alpha_1 V_{2m-1}^0 + \alpha_2 V_{2m}^0 + \alpha_3 V_{2m+1}^0 + \alpha_4 V_{2m+2}^0 \quad (3.2.23)$$

Similarly, the second level Daub4 scaling signals are defined by

$$V_m^2 = \alpha_1 V_{2m-1}^1 + \alpha_2 V_{2m}^1 + \alpha_3 V_{2m+1}^1 + \alpha_4 V_{2m+2}^1 \quad (3.2.24)$$

All other levels of Daub4 scaling signals are defined in a similar fashion.

Let the *wavelet* numbers $\beta_1, \beta_2, \beta_3, \beta_4$ be defined by

$$\beta_1 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \beta_2 = \frac{\sqrt{3}-3}{4\sqrt{2}}, \beta_3 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \beta_4 = \frac{-1-\sqrt{3}}{4\sqrt{2}} \quad (3.2.25)$$

Notice that the wavelet numbers are related to the scaling numbers by the equations: $\beta_1 = \alpha_4$, $\beta_2 = -\alpha_3$, $\beta_3 = \alpha_2$, $\beta_4 = -\alpha_1$. Using these wavelet numbers, the 1-level Daub4 wavelets $W_1^1, W_2^1, ..., W_{N/2}^1$ are defined by

$$\left.\begin{aligned} W_1^1 &= (\beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots\ldots\ldots, 0) \\ W_2^1 &= (0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots\ldots\ldots, 0) \\ W_3^1 &= (0, 0, 0, 0, \beta_1, \beta_2, \beta_3, \beta_4, 0, 0, \ldots\ldots\ldots, 0) \\ &\vdots \\ W_{\frac{N}{2}-1}^1 &= (0, 0, \ldots\ldots\ldots, 0, \beta_1, \beta_2, \beta_3, \beta_4) \\ W_{\frac{N}{2}}^1 &= (\beta_3, \beta_4, 0, 0, \ldots\ldots\ldots, 0, \beta_1, \beta_2) \end{aligned}\right\} \tag{3.2.26}$$

The 1-level Daub4 wavelets satisfy

$$W_m^1 = \beta_1 V_{2m-1}^0 + \beta_2 V_{2m}^0 + \beta_3 V_{2m+1}^0 + \beta_4 V_{2m+2}^0 \tag{3.2.27}$$

Similarly, the 2-level Daub4 wavelets are defined by

$$W_m^2 = \beta_1 V_{2m-1}^1 + \beta_2 V_{2m}^1 + \beta_3 V_{2m+1}^1 + \beta_4 V_{2m+2}^1 \tag{3.2.28}$$

All other levels of Daub4 wavelets are defined in a similar fashion.

3.2.4 Application of Wavelets in De-noising Data.—As mentioned, wavelets are being applied in a variety of applications. A few of the applications are in computer and human vision, FBI fingerprint compression, denoising of noisy data, detecting self-similar behavior in a time-series, and musical tones. In recent years researchers started using wavelet theory as a technique for damage detection. But in this study, it is used as tool for denoising the measurement data.

In this study, wavelets are utilized in the signal feature reconstruction from the noisy data. When we decompose a signal using wavelets, we use filters that serve as averaging filters and others that produce details. Some of the resulting wavelet coefficients correspond to details in the signal. If the details are small, they might be omitted without affecting the main features of the signal. The idea of thresholding is applied to set to zero all coefficients that are less than a particular threshold. These coefficients are used in an inverse wavelet transformation to reconstruct the signal.

There are two types of thresholding used in denoising, i.e., hard thresholding and soft thresholding. figures 3.1 and 3.2 shows the hard and the soft thresholdings, respectively. Hard thresholding is not a continuous function, so it will exaggerate the gaps between significant and insignificant values. Soft thresholding is a simple modification of hard thresholding such that is becomes a continuous function which as a result, it does not exaggerate the gap between significant and insignificant values. In general, the choice of thresholding depends on the type of application.

(1) Hard Thresholding

$$H(x) = \left\{\begin{aligned} x \quad & \text{if } |x| > T \\ 0 \quad & \text{if } |x| < T \end{aligned}\right\} \tag{3.2.29}$$

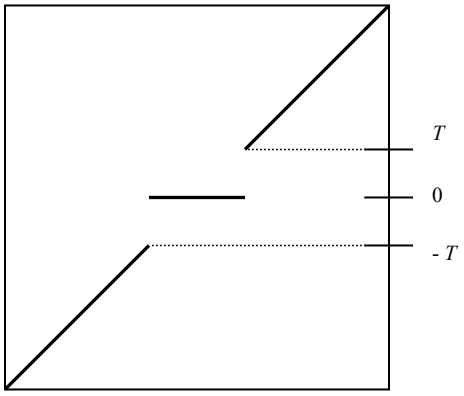


Figure 3.1.—Hard thresholding.

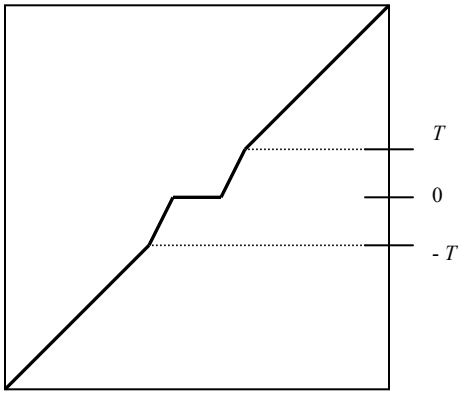


Figure 3.2.—Soft thresholding.

(2) Soft Thresholding

$$S(x) = \begin{cases} x & \text{if } |x| > T \\ 2x - T & \text{if } T/2 \le x < T \\ T + 2x & \text{if } -T < x \le -T/2 \\ 0 & \text{if } |x| < T \end{cases} \qquad (3.2.30)$$

Let us consider a noisy signal, which is taken from the wavelet toolbox demos and it is called as noisy blocks, as an example to see the effect of denoising by using a fifth level Daub4 wavelet found in the Wavelet Toolbox in Matlab [ ]. The Daub4 wavelet is denoted by "db2" in wavelet toolbox. Here, different values of soft thresholding, T, are used. Figures 3.3 to 3.6 show the effect of the various values of the threshold, T. As expected, we can observe more reduction of the noise in the signal by increasing the value of T. The values of the T need to be "optimized" in order to retain the significant values of the original signal. Figure 3.7 shows the details/differences of the noised signal with a threshold, $T = 4.219$. The dotted line in each level of details (d_1 to d_5) represents the value of each threshold in which the denoising coefficients below that threshold value are set to zero.

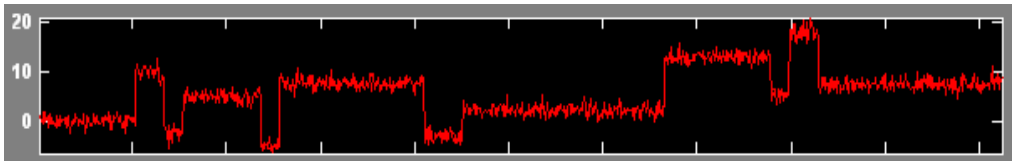


Figure 3.3.—Signal with the noise.

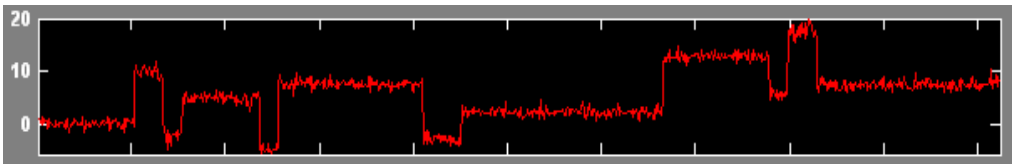


Figure 3.4.—Denoised signal using threshold = 0.5.

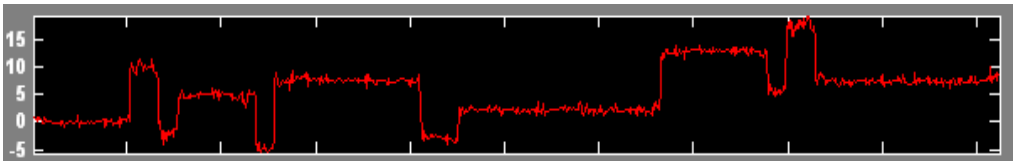


Figure 3.5.—Denoised signal using threshold = 1.

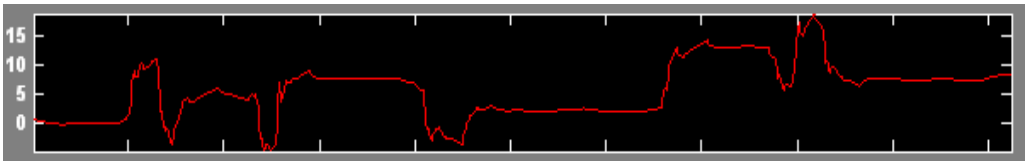


Figure 3.6.—Denoised signal using threshold = 4.219.

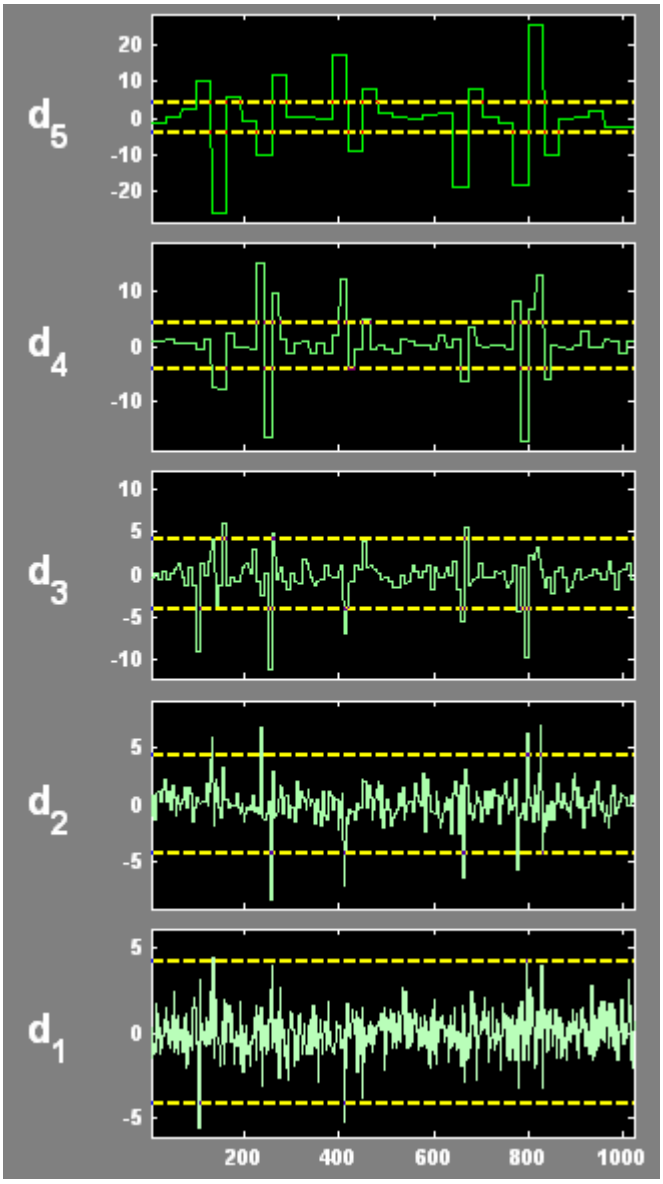
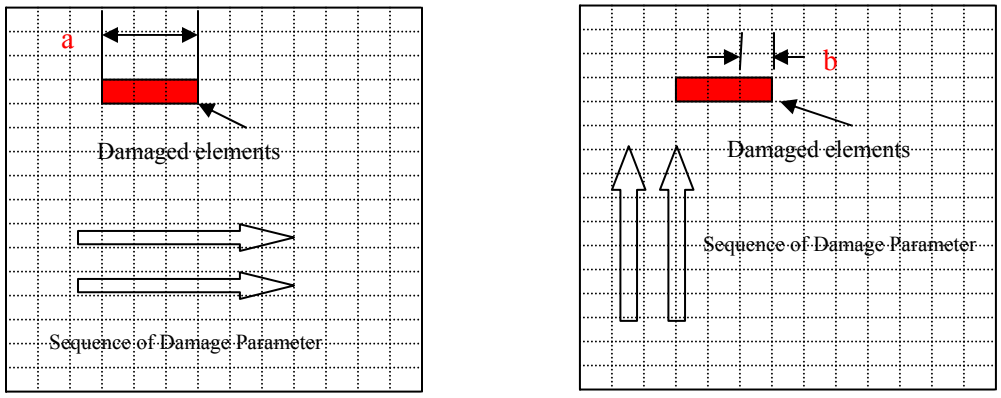


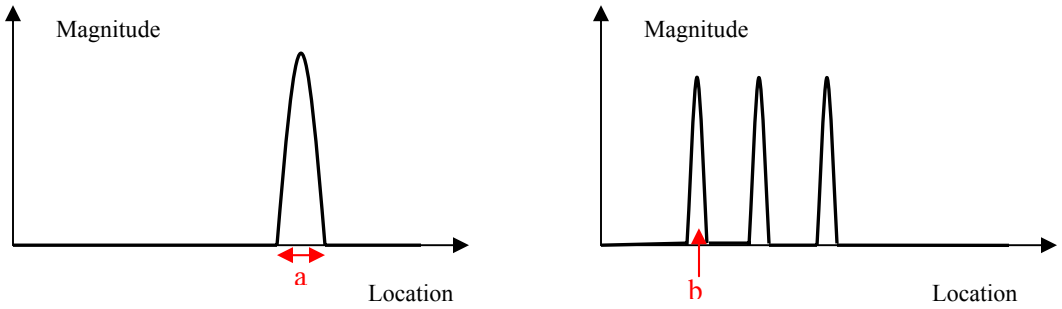
Figure 3.7.—Details/differences of noised signal with threshold = 4.219.

3.2.5 Specific Application of Wavelets in Global Damage Detection.—The signal considered in the above is an example discrete signal in which a scalar quantity is a function of time (as in most typical applications of wavelets). In this present research, the damage parameter values (a scalar) are interpreted as a discrete "signal", which is not a function of time, but instead is a function of position. That is, the scalar damage parameter is calculated at various points in a structure, which for the case of a flat plate have a specific position (x,y) value associated with it. Thus it is necessary to convert the "two-dimensional" (spatial) array of damage parameter values to a one-dimensional array of values. To achieve this, the values of the damage parameter are processed in a sequential manner over the area of the plate. Figure 3.8(a) shows the typical sequence in which the damage parameter values are read. It is very important to note that when forming the damage parameter signal, the sequence in which the values are read is of prime consideration.



(a) (b)

Figure 3.8.—A plate with a mesh size of 13 by 16, wherein three elements are damaged (a) Sequence of damage parameter consideration is "left to right" (b) Sequence of damage parameter consideration is "bottom to top."



(a) (b)

Figure 3.9.—Damage parameter signal (a) Sequence from "left to right" (b) Sequence from "bottom to top."

Figure 3.8 shows a plate with a mesh size of 13 by 16. In the total of 208 elements, three elements are damaged at a location, i.e., treating as single damage in the plate. When the damage parameter is generated, damaged elements will have the maximum values. When we form a discrete signal by considering the damage parameter values from "left to right" as in figure 3.8(a), then the signal will have peak at single location as in figure 3.9(a), since three damaged elements are next to each other in this sequence. But, if we consider from "bottom to top" as in figure 3.8(b), then the signal will have peaks at three locations as in figure 3.9(b), since three damaged elements are not next to each other in this sequence. In both cases, there is single damage. Therefore, in the signal feature reconstruction, care has to be taken on the sequence of damage parameter when forming a discrete signal.

In the following sections of this report, a simulation case study is performed pertaining to signal feature reconstruction using the discrete wavelet transforms outlined above. The wavelet toolbox in Matlab is used for all simulations and the results of this case study are presented in chapter V.

Chapter IV
Computational Tools

4.1 General

In the general engineering community, the use of computational simulation software has become an obvious solution for many of the challenging engineering problems. In engineering, the design/analysis phase requires an iterative process in order to obtain the optimal design. Computational simulation software can handle this iterative process very efficiently, so an engineer can concentrate on other major issues of the work. In this context, many software tools have been developed for engineering applications. Software tools such as ABAQUS, ANSYS, LS-Dyna, etc. are examples of widely used general purpose finite element based structural analysis programs. Other tools such as Matlab, Mathematica, etc. are useful for such topics as signal processing, optimization, symbolic derivation, etc. All of these tools serve to increase the productivity/efficiency of the engineer.

With the above points in mind, an efficient computational tool to assist in structural damage detection is developed in this research. A computational tool, which will be referred to as the D-Tector, previously developed in (refs. 67, 68, and 85), is used for the global (structural) damage detection. With D-Tector serving as the central software component, other software packages such as ABAQUS, Visual C++, Matlab, and Wavelet Toolbox are combined in a computationally efficient manner to allow for the systematic study to be presented in subsequent chapters. Figure 4.1 describes how these tools were utilized. The following sections present a detailed explanation of the implementation of these tools in this research.

4.2 D-Tector for the Global Damage Detection

4.2.1 Introduction.—Observing the necessary usefulness of having a capability for structural diagnosis and prognosis, this present research study focuses on using a finite element based algorithm for the damage detection and localization of a structural member. Primarily, this study concentrates on the development of a robust scheme for the global damage detection which utilizes as a central component in the D-Tector software.

The main purpose of D-Tector is the detection, location and determination of the approximate size of defect/defects in a structural member, using the displacements measured during the experiments. Considering the use of the latest experimental techniques, viz. the Electronic Speckle Pattern Interferometry (ESPI) method of producing interferograms, D-Tector contains a feature to convert the experimental file written in binary format to ASCII format and ultimately to the input file format necessary for the damage detection core. This computational tool contains a series of programs written in Fortran and C and a graphical interface, called Eview, for post processing. Eview is based upon various C Graphics libraries and is used visually interpret/extract the damage.

4.2.2 Structure of the D-Tector.—A program's efficiency is directly related to the structure of any computational tool. The D-Tector tool has three major sections: Preliminary Processing, Damage Detection Core, and Post Processing. Each section contains its own subroutines with a special purpose.

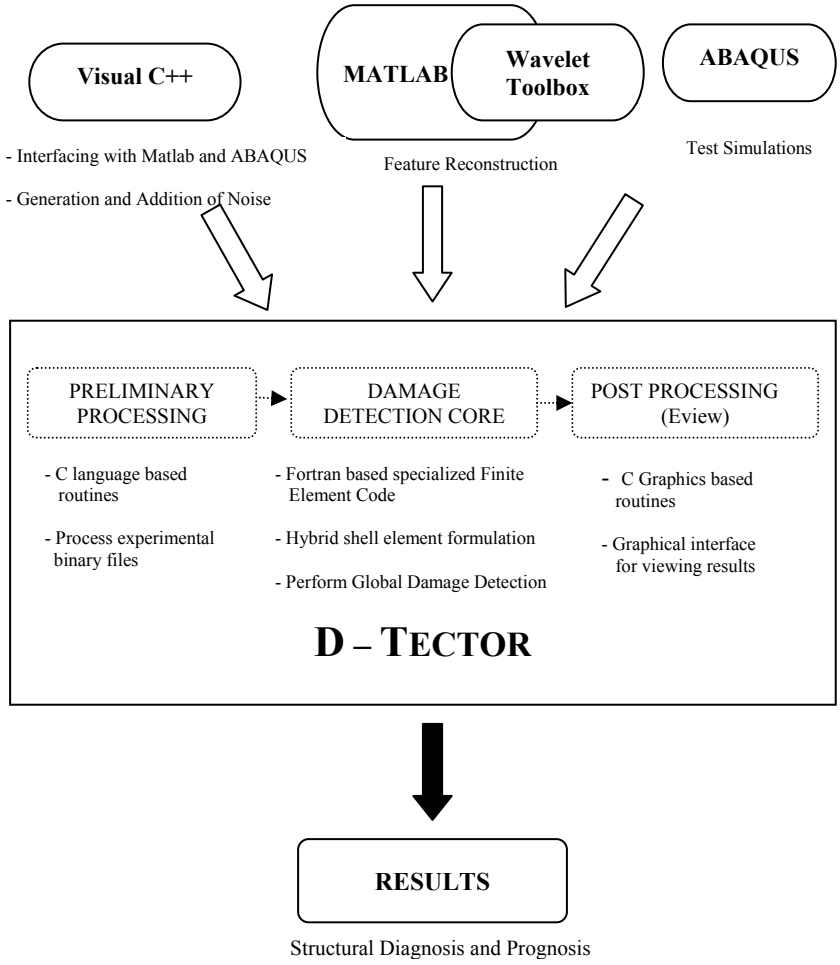


Figure 4.1.—Computational tools for global damage detection.

Figure 4.2.—Structure of the D-Tector.

The developed NDE scheme for global damage detection is contained in the Damage Detection Core section. Therefore, this section will handle the necessary computations for the detection. As written, the damage detection core can understand only specific formats of the input, but the experimental files may be in a binary format which is not compatible. The process to modify the experimental files into a format, which is compatible to the damage detection core, is undertaken by the Preliminary Processing section. The format of this input file has been designed in such a way that the user can generate it manually, if the frequency of vibrations and the displacements of the structure at specific locations are known. In this case the so-called preliminary processing is not needed. However, since many experimental techniques today are conducted using digital computers for data acquisition, and which may be in binary format, the pre-processing capabilities are required. Similarly, the post-processing section is for viewing the files

generated by the damage detection core for the so-called “feature extraction.” By viewing the files with Eview, the user can detect and locate and measure the approximate size of the damage. Eview also has the provision for plotting the mode shapes of the structure.

4.2.3 Algorithm of the D-Tector.—The initial stage of the D-Tector is the preliminary processing. It contains four applications viz. Lazout2surfit.exe, Surfit.exe, Pix2ev.exe, and Pix2evNonorm.exe. The input file for the damage detection core will be generated from the experimental file in this section. The figure 4.3 illustrates the series of the applications utilized.

A typical laser holographic testing experimental setup that measures the modal response, which is used for the global damage detection, generates a binary output file with .out extension. To retrieve the holographic data, i.e., to convert the binary file to ASCII format, the Lazout2surfit.exe program is utilized. This program takes the experimental output file as input and coverts into the surfit file, which is in ASCII format. The retrieved holographic data in the surfit file appears as an array of pixel values. Also, reduction of the data may be required if large volumes of data is produced for the efficiency of the analysis. The surfit.exe program is used to translate and truncate the data, i.e., for ignoring particular regions of data. This code fits the translated data and samples the data at a resolution that the damage detection core can use. The translated data from the surfit.exe is stored in the .xyz file. The .xyz file consists of number of columns, which indicates the location and the corresponding intensity at that location of the plate according to the dimensions of the plate. The first column indicates the *x*-coordinates, second column indicates the *y*-coordinates and the third column indicates the intensity/value at that particular location. The sequence continues until the end of the plate.

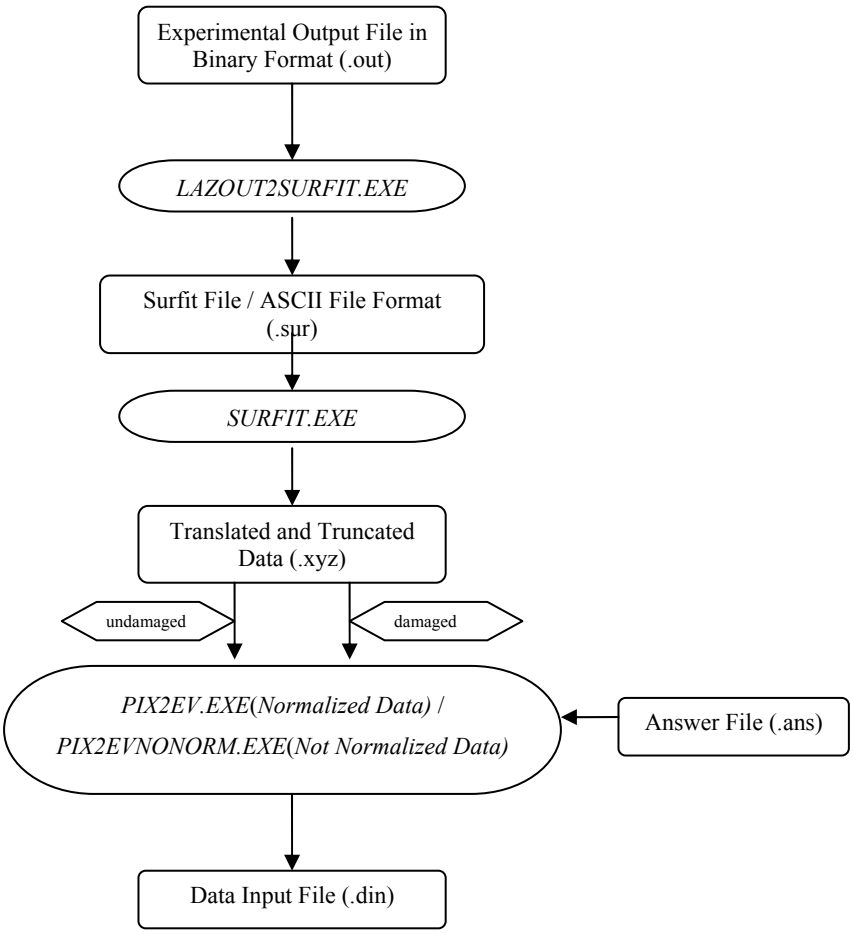


Figure 4.3.—Flow chart for preliminary process.

The damage detection core accepts a specific format of the input file. Pix2ev.exe generates the damage detection core input decks with normalized displacements by taking the ud.xyz (undamaged) file and the d.xyz (damaged) file as input. Also, there are few variables that need to be input in addition to the damaged and undamaged files, such as the frequencies, values for the mesh size, and the thickness of the plate. There is the option to put this data in an "answer" file so that the program may be run in a "batch mode" as opposed to interactively to decrease the burden on the user. All of the input variables are in the answer file, so the user only has to change the damaged and undamaged frequencies, values for mesh size, and the plate thickness for each analysis. Pix2ev then creates a data input (.din) file, which will be used as the input file to the damage detection core. Figure 4.4 illustrates the major segments of the typical input file. The data input file consists of the degrees of freedom, nodal and element definitions, boundary conditions, frequencies of vibration of the undamaged and damaged plates, and the undamaged and damaged displacements.

The displacements are normalized when pix2ev.exe is adopted for developing the data input file. Pix2evNonorm.exe is also develops the data input file (.din) but without normalizing the undamaged and damaged displacements. If you want to consider the actual values instead of normalized values in the detection this application is needed.

The data input file generated by the preliminary processing will now be in a format that is usable to the damage detection core. At this point, the data will be sent to the damage detection core section where the damage computations are performed. The damage detection core section generates the files in such a way that the user can view in Eview the location and approximate size of the damage. This section contains three main applications for the D-Tector, viz., asg_dam.exe, asg_get_dam_vector.exe, and evdelta.exe.

Figure 4.5 emphasizes the sequential list of applications in the Damage Detection Core section. The asg_dam.exe program is the heart of the D-Tector. The major formulation for the damage detection is coded in this application. It has the ability to perform the static and dynamic analyses and the damage parameter computations. Specifically, this code is finite element based. Since this study focuses on plate-type structures, a 3-D shell element with five degrees of freedom (3 translations and 2 rotations) is used primarily. For generality, the code also has ability to work with four types of elements viz. truss, beam, curved beam, and shell elements. Asg_dam.exe produces the output (.dot) file. This data output file is very large and very significant in the detection process. This file contains nodal and element definitions, displacements, stress calculations, damage parameter calculations, load case data, and total system data i.e., information about the number of equations, the number of matrix elements and the bandwidth. By simply reading large "raw" data output file, it is very difficult to extract the significant damage features. Thus, the asg_get_dam_vector.exe program is used to extract the specific information required to graphically view the results. It generates five files viz. raw damage parameter (.rdp), smooth damage parameter (.sdp), damage vector (.dav), undamaged mode shape (.uds), damage mode shape (.dds). The .rdp file contains the raw nodal damage parameter vector definitions and element data damage

Nodal Definitions

```
Phase1 fea_test for documentation purpose, dimension 7"x 5.25"x0.125", mesh 40 x 40
5    61   1  1  1  1  1  1  1
0      0
0
       1   0     0.000000    0.000000   -0.062500    1    0.125000    0
                 0.000000    0.000000    0.062500
       6   0     7.000000    0.000000   -0.062500    0    0.125000    0
                 7.000000    0.000000    0.062500
       7   0     0.000000    0.000000    0.000000    1    0.125000    0
                 0.000000    0.000000    0.000000
      11   0     0.000000    0.000000    0.000000    0    0.125000    0
                 0.000000    0.000000    0.000000
      12   0     0.000000    1.050000   -0.062500    1    0.125000    0
                 0.000000    1.050000    0.062500
      17   0     7.000000    1.050000   -0.062500    0    0.125000    0
                 7.000000    1.050000    0.062500
      18   0     0.000000    0.000000    0.000000    1    0.125000    0
                 0.000000    0.000000    0.000000
      22   0     0.000000    0.000000    0.000000    0    0.125000    0
                 0.000000    0.000000    0.000000
```

Boundary Conditions and Material Definitions

```
1          1   1   0   0   0   1
60         1   1   0   0   0   0
1          1   1   1   1   1   1
6          1   1   1   1   1   0
61         1   1   0   0   0   0
1        0   0   0
5        0.0
4        25  2   5   3   2   1   1   0   0   0   1   0   0   0
1             7.67E-04
2.96E+07        0.3   0.0   0.0
2             7.67E-04
1.00936+07      0.3   0.0   0.0
```

Element Definitions

```
  1    5   1   1
  1        2       13       12        7   1  0.0
  5    5   1   0
  5        6       17       16       11   1  0.0
  6    5   1   1
 12       13       24       23       18   1  0.0
 10    5   1   0
 16       17       28       27       22   1  0.0
 11    5   1   1
 23       24       35       34       29   1  0.0
 15    5   1   0
 27       28       39       38       33   1  0.0
 16    5   1   1
```

Undamaged Frequency and displacements

```
10025315020.326000
 86    1       3       0.00801389
 87    1       3       0.00735845
 88    1       3       0.00677046
 89    1       3       0.0062266
 90    1       3       0.00571156
 91    1       3       0.00521458
 92    1       3       0.0047283
 93    1       3       0.00424903
 94    1       3       0.00377417
 95    1       3       0.00330136
 96    1       3       0.00282931
 97    1       3       0.00235715
 98    1       3       0.00188609
```

Damaged Frequency and displacements

```
0
52681441.198237
  84    1       3       0.00952708
  85    1       3       0.00870306
  86    1       3       0.00795856
  87    1       3       0.00730549
  88    1       3       0.00671731
  89    1       3       0.00617249
  90    1       3       0.00565688
  91    1       3       0.00515783
  92    1       3       0.00467108
  93    1       3       0.00418977
  94    1       3       0.00371273
  95    1       3       0.00323724
  96    1       3       0.00276415
```

Figure 4.4.—Major segments in data input file.

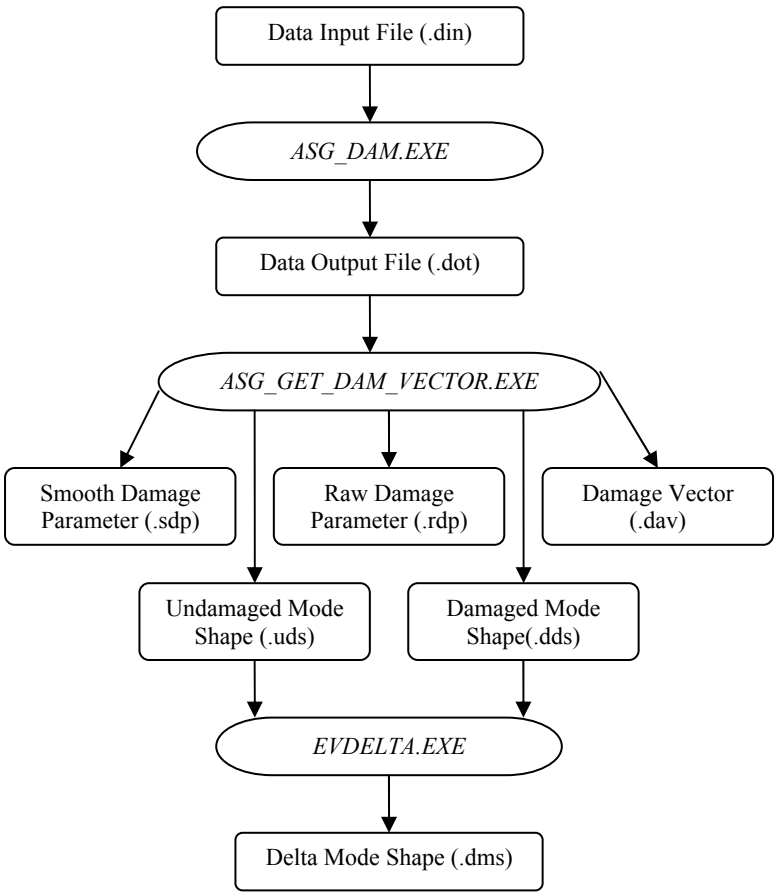


Figure 4.5.—Flow chart for damage detection core.

parameter definitions where as the .sdp file contains the smoothened nodal damage parameter vector definitions and element data damage parameter definitions. The nodal damage parameter values are per-element values that have been area-weighted at the nodes. The .dav file contains the damage force vectors. These are calculated based on the energy vectors in the structural member due to the vibrations. The undamaged and the damaged mode shape files contain the displacements extracted from the data output file. All of these files can be viewed in the post-processing section.

The damage detection core has the option to deal with either dynamic or static analysis. Thus, the asg_dam.exe program also has a provision to accept different formats of the data input file. We can feed the displacements which are generated experimentally or analytically. Analytical data can be generated by using any finite element analysis software. If we provide the displacements to the damage detection core then D-Tector uses the series of programs listed in the figure 4.5. In this case, the analysis is treated as static. The damage detection core can generate the dynamic response for the purpose of simulations. For the dynamic case the input file is different from the static case. When testing analytically we must provide the number of vibration modes with damaged element/elements definitions since the displacements are unknown. The code will generate the displacements according to the out-of-plane or in-plane action of the analysis. If performing purely a static analysis the user should provide the applied force definitions on the nodes. The damaged elements can be defined by the degradation of the Modulus of Elasticity. The damage calculations are performed and written in the data output file. In the analytical case to acquire the mode shapes a separate program, asg_get_mode_vector.exe, was developed. It generates undamaged and damaged mode shape files with .mdv extension.

The displacements of the undamaged and damaged plates are different in value. This difference is because of the presence of the damage. Therefore, if we evaluate the difference between both displacements, we identify that the delta mode shape is completely different than the damaged and undamaged mode shapes. This delta mode shape is a measure of the accuracy of the experimental technique. The evdelta.exe program is developed to serve this purpose. The undamaged and damaged mode shapes generated either by the analytical or by the experimental files will be sent to the evdelta.exe as input to get the difference. This file has the .dms (delta mode shape) extension.

The final and most important section in the D-Tector is post-processing. The files generated by the D-Tector contains huge amount of data and identifying the significant damage values is a difficult task. A viewer is necessary to extract the feature of the damage in a proper pattern. In conjunction to this, Eview is developed. Figure 4.6 shows the Eview window. The files generated by the damage detection core can be viewed in Eview to locate the damage. It has developed using various C graphics libraries and has the feature of NURB sampling fidelity for interpolation, which allows the better viewing of the contours to help in localizing the damage. The mesh of the member can be viewed with nodal and element numbers displayed.

The E-view has provided six different views to extract feature of the damage for specific purposes. By selecting the first option user can view the picture in black and white shades according to the values. The second option differentiates the positive and negative values. The contours are plotted in the colors, positive is white and negative is red, for easy differentiation. Positive values are indicated in white to black shade and negative values are indicated in light red to dark red shade in the picture. The third option uses the color spectrum to indicate the values in the picture. The first three options use the NURB sample fidelity and viewed as contours. Fidelity can be increased to sharpen the picture. The fourth option shows the magnitude per element. The fifth option is to view the values of the files in the vector field. There is a provision made to change the vector view threshold to hide the unwanted vectors to extract the damage. The vectors that are below the threshold value are erased from the picture to indicate the damage location more clearly. There is also a provision is made to increase the scale of the vectors. The final option is to

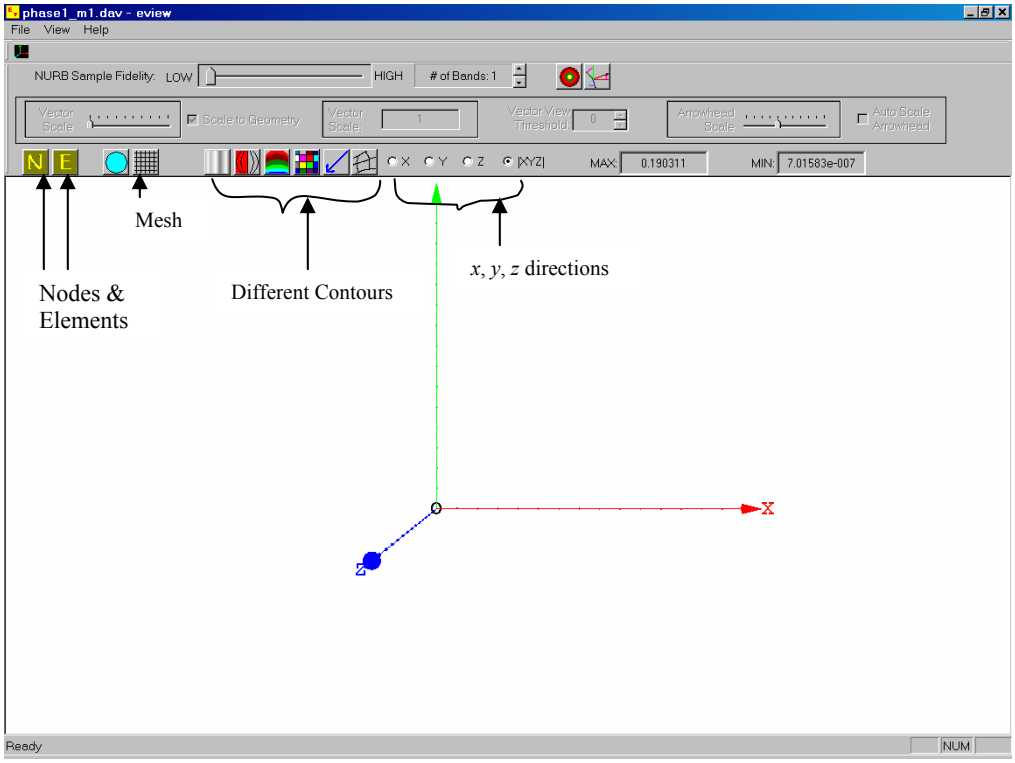


Figure 4.6.—Eview window.

view the files in a meshed pattern using rainbow colors to indicate the values. Depending upon the type of file, the user can choose the appropriate contour to extract the required feature. The figures presented to describe the results of this study were produced by the E-View. Different types of contours produced by the E-View are described in the further chapters.

The features that can be extracted from the Eview are presented from figures 4.7 to 4.12. Here, Mode-5 is taken as example to show the features.

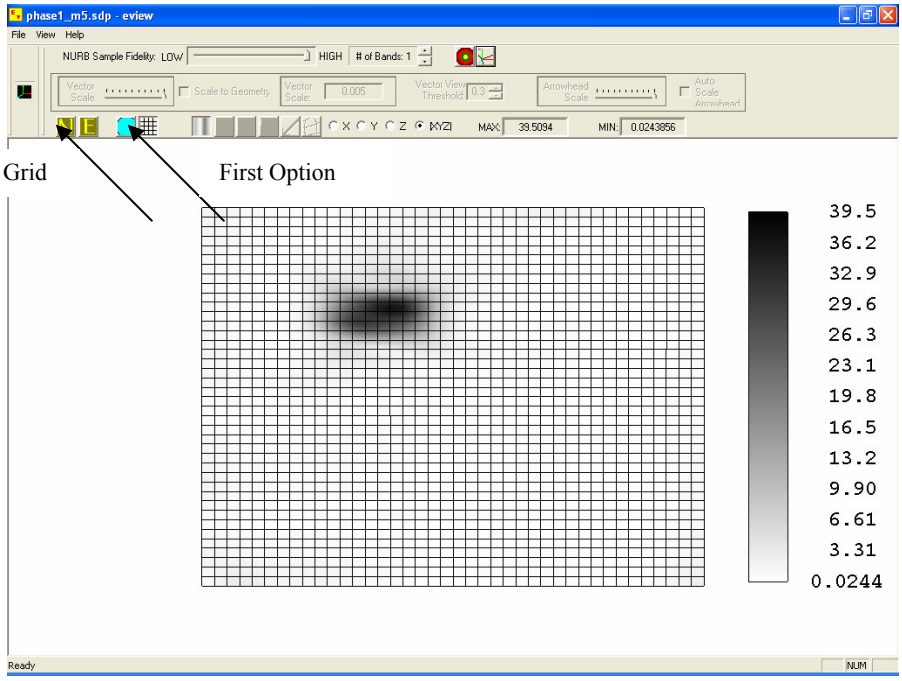


Figure 4.7.—Damage parameter contour using black and white intensities.

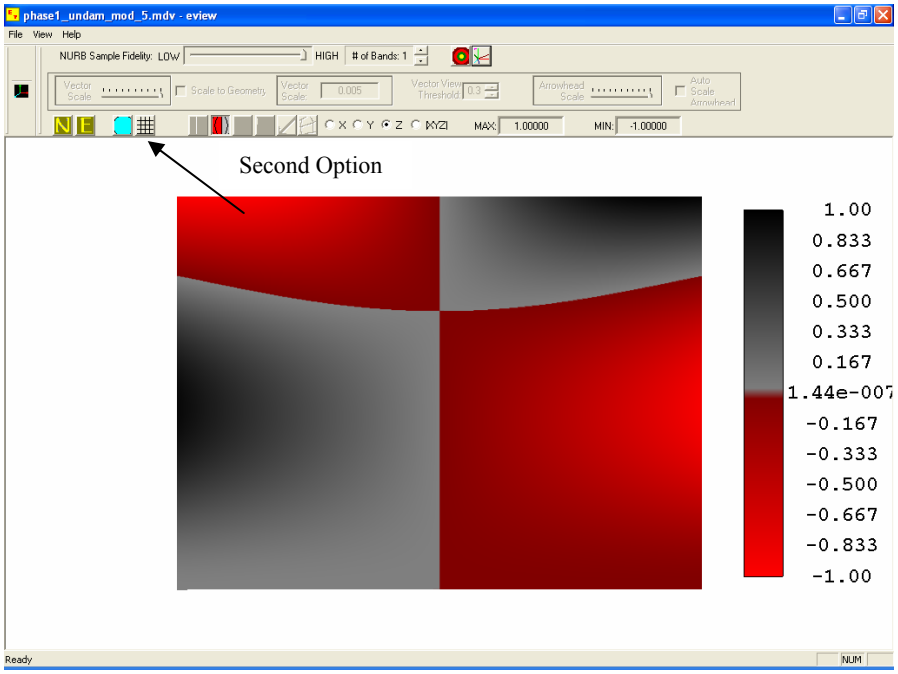


Figure 4.8.—Mode Shape using different colors for positive and negative directions.

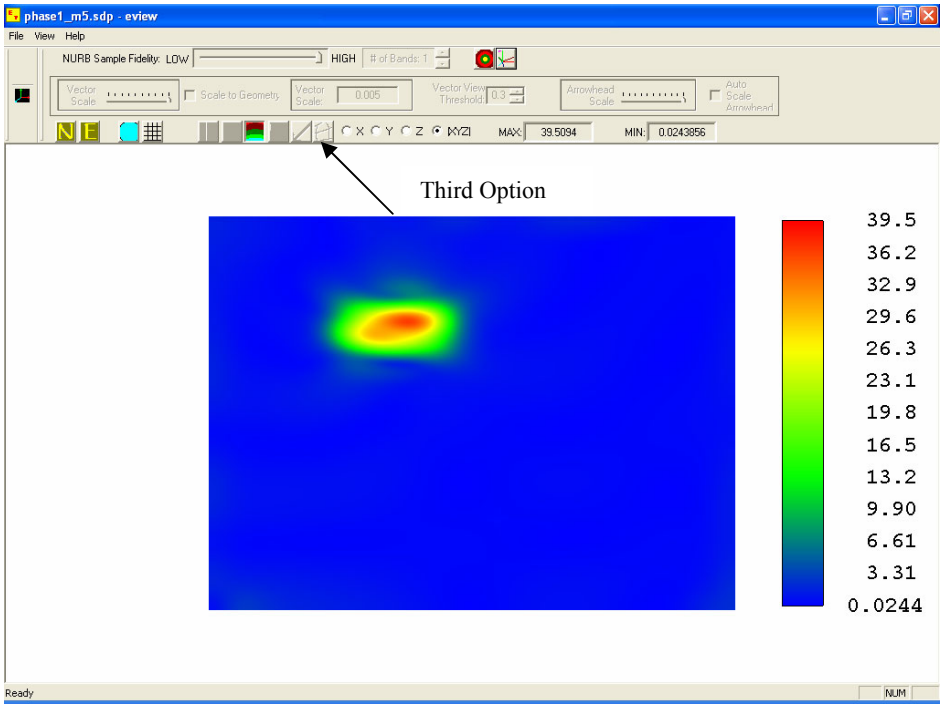


Figure 4.9.—Damage parameter contour using rainbow colors with NURB fidelity.

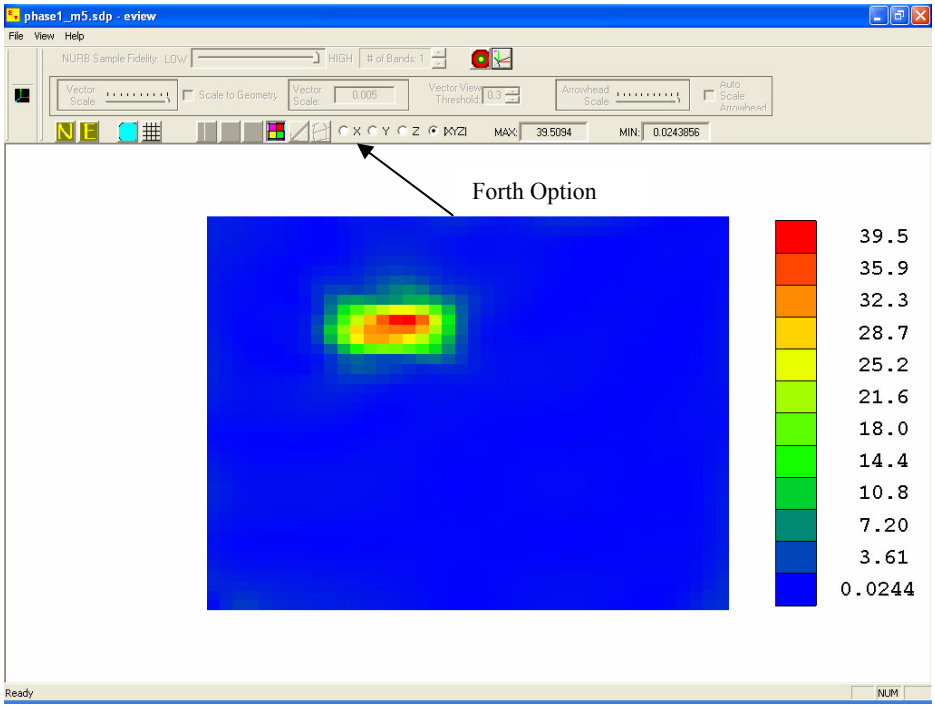


Figure 4.10.—Damage parameter contour using rainbow colors without NURB fidelity.

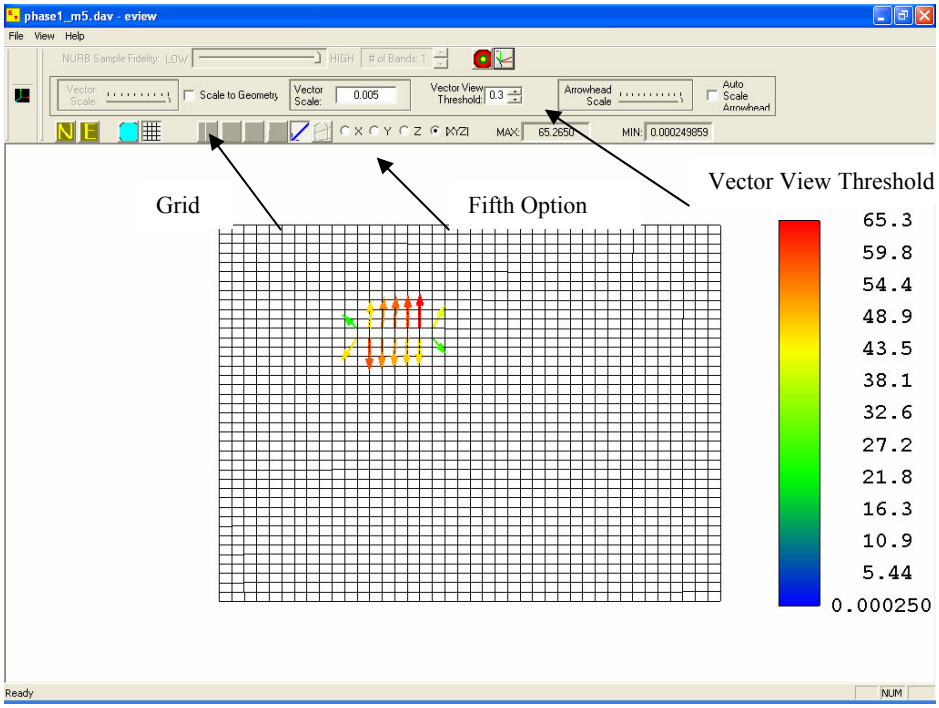


Figure 4.11.—Damage vectors.

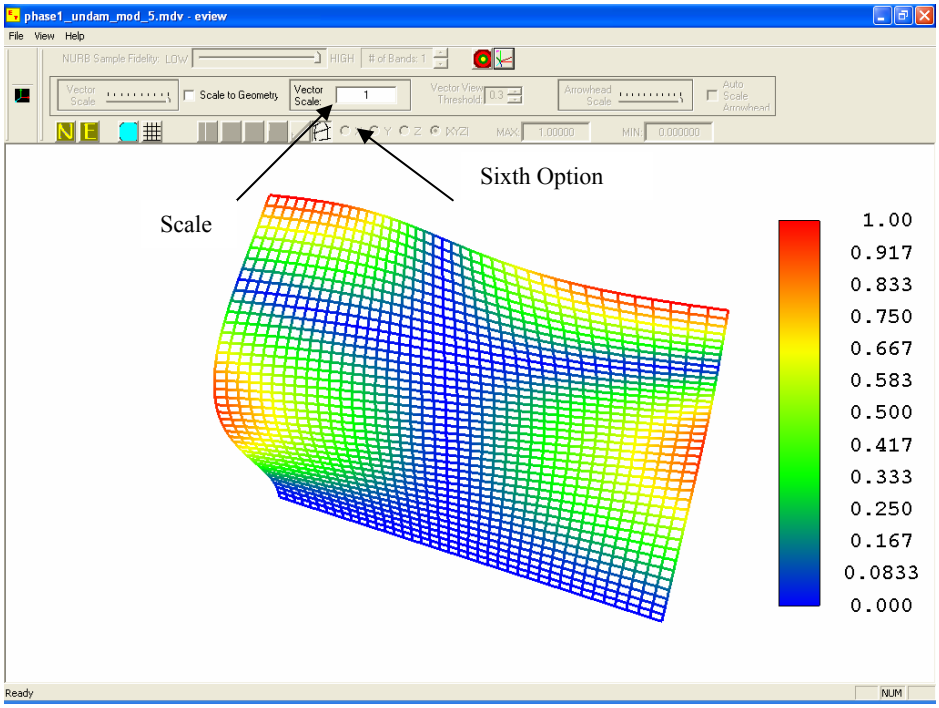


Figure 4.12.—Mode shape.

In summary, the main computational tool for detecting the damage is D-tector. Up to this point, we have not considered the possibility of noise in the experimental measurements. Since the primary focus of this research is for extracting the damage in the presence of noisy data measurements, the following sections will describe the utilization of other software tools necessary to process the data.

4.3 Visual C++ for Connectivity and to Study the Effect of Noise

In reality, obtaining noise free measurements are very difficult. Therefore, assessment of the accuracy of the measurement data to noise is needed for a robust detection scheme. To simulate the actual laboratory environment, computer generated random numbers are added to the data sets. Here, Visual C++ is utilized to generate and add the noise in which the "rand( )" function in the Visual C++ library is used to generate the random numbers. It also used to connect the D-Tector to other tools like ABAQUS and Matlab. The main applications developed by the Visual C++ are GK_Noise_Damage_Parameter.exe, GK_InputFromAbaqus.exe, GK_GenNoise.exe, and GK_AddNoise.exe. GK_Noise_Damage_Parameter.exe connects the D-Tector to Matlab and loads the noised damage parameter values to Matlab and extracts the "denoised" values from Matlab. GK_InputFromAbaqus.exe connects the D-Tector to ABAQUS and will extract the mode shapes generated by the ABAQUS modal analysis and creates the data input file in form that is compatible to D-Tector. GK_GenNoise.exe generates the noise, random numbers, and writes into a file. GK_AddNoise.exe adds the noise to the data sets in the data input file by using already generated noise files from GK_GenNoise.exe. The source codes of the developed applications by Visual C++ are presented in the appendix.

4.4 ABAQUS for Detection Scheme

ABAQUS is a general purpose commercial Finite Element Analysis software for linear and nonlinear analysis. It is mainly used to perform modal analysis. Mode shapes/displacements are saved in a file from the ABAQUS viewer. This data is used in D-Tector or wavelet toolbox wherever required.

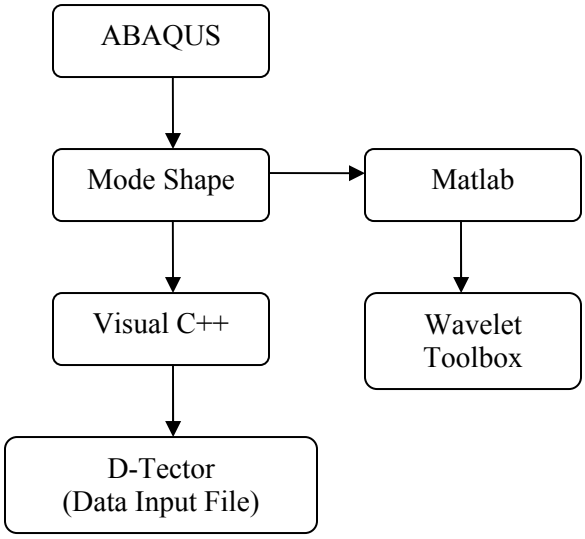


Figure 4.13.—ABAQUS and Matlab in global damage detection.

4.5 Matlab for Detection Scheme

The Matlab is also commercial multipurpose numerical analysis software for engineers. It is useful for coding and plotting mathematical functions, and to perform other mathematical simulations/calculations. In addition, Matlab can connect to special purpose toolboxes viz., Signal Processing, Image Processing, Wavelet, Curve fitting, Control Systems, Fuzzy Logic Toolboxs, etc. In the present research, Matlab is used to connect to the Wavelet Toolbox which will be used to "denoise" the measurement data.

4.6 Wavelet Toolbox for Signal – Feature Reconstruction

In the last two decades, extensive research has focused on the practical application of wavelet theory to a variety of engineering problems. According to the theoretical background described in Chapter III, researchers have developed number of wavelets. The most popular wavelets are assembled into one computational tool, called Wavelet Toolbox developed by Mathworks, Inc. This toolbox contains 15 different wavelet families with a Graphical User Interface (GUI). It contains 1-D and 2-D wavelet analysis with Discrete, Continuous and Stationary Wavelet Transforms. The main applications of the wavelet toolbox are Denoise and Compression of signals and images. In this study, it is used for feature reconstruction of noisy data. The Daubechies2 (Db2), which is referred as Daub4 in the literature, wavelet is used from the predefined wavelets in wavelet toolbox.

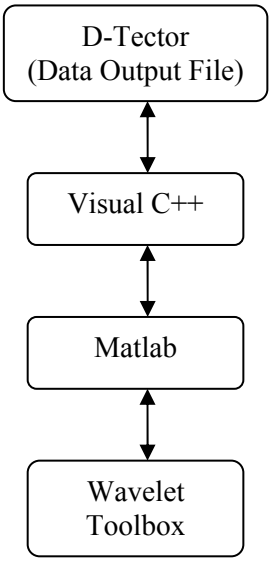


Figure 4.14.—Matlab and Wavelet Toolbox for
feature reconstruction in global damage detection.

Table 4.1 summarizes the purpose of the described computational tools in this chapter. The results of simulations performed and the application of these computational tools are discussed in chapter V.

TABLE 4.1.—COMPUTATION TOOLS

Computational tool	Purpose in general	Purpose in this research
D-Tector	Global damage detection	Global damage detection
Visual C++	Programming	Connecting D-Tector to other tools and to study the noise
ABAQUS	Finite element analysis	Performed required simulations
Matlab	Coding and plotting software for engineers	Used to connect Wavelet Toolbox
Wavelet Toolbox	Wavelet analysis	Signal – feature reconstruction

Chapter V
Simulations and Results for Case and Comparitive Studies

5.1 Introduction

Damage detection is a challenging problem that is under vigorous investigation by numerous research groups using variety of analytical and experimental techniques. The damage detection procedure for a structure depends upon the level and extent of the damage, available knowledge concerning the ambient dynamic environment, sophistication of available computing resources, complexity of the detection scheme, and depth of knowledge concerning the failure modes of the structure.

One of the most important criteria to be fulfilled in any damage detection scheme based on a theoretical/computational model is the robustness of the method to real applications; i.e., using data from a laboratory setup or field measurements. In particular, of utmost importance for the effectiveness of the method is its ability to determine the location and severity of the damage when the experimental measurements (e.g., strains, deflections, rotations, etc.), which may contain various environmental factors and human errors, are provided. Simulations have been done to study the effectiveness of the proposed detection scheme based on the following considerations. Damage in a structure can appear as weakened material (e.g., in low cyclic fatigue), line cracks (e.g., high cyclic fatigue), or corrosion. If we consider the sensor network in an experimental setup, it can be an extensive network or coarser network. The detection scheme should be able to determine the location and severity of any type of failure in the structure even with a coarse sensor network. And also scheme should withstand some amount of noise. Along with the above simulations, feature reconstruction methods are considered to reduce the noise intensity in the measurement data. Wavelet transforms are utilized for feature reconstruction in this study. The background on the simulations is described in detail in the following subsections.

5.1.1 Failure Modes/Type of Damage.—Keeping an eye on the real time environment, simulations are performed to complete this research. In damage detection, failure modes of a structure are one of the important factors. The types of common failure modes are classified as Low Cycle Fatigue, High Cycle Fatigue, and Corrosion. The damage in a structure can occur after a low number of cyclic loadings in the Low Cyclic Fatigue failure mode. This type of damage is simulated by a sudden degradation of the Modulus of Elasticity of element/elements in the finite element model. If the damage occurs in a structure after a high number of cyclic loadings then it falls under the High Cyclic Fatigue failure mode. This type of damage is simulated as a line crack by creating multiple nodes at the same geometric location and changing the nodal connectivity. Corrosion of the material is also a type of damage in the structure. This type of damage occurs in very old structures. When the corrosion of a structure starts, then we can observe tiny holes. This is simulated by eroding an element from the structure's geometry. The finer the mesh, smaller the hole size is. Therefore a very fine mesh is generated to simulate a tiny hole in the structure.

5.1.2 Sensor Network in Measurements.—Density of the sensor network can affect the detection scheme. Arrangement of the sensors for the measurements is the most important factor. The extensive/fine network assumes an ideal, "unlimited" ability to place and measure response at all regions in the structural domain. However, from a practical viewpoint, the case of coarse networks, where no direct measurements are made at or very near to the ("presumably unknown") locations of the damage, are very crucial in judging the potential of any damage detection methodology. To emphasize this point, all selected coarse networks completely avoid any measurements at or very near to areas of defects. To study the affect of sensor network, initially, a 40 by 40 mesh is considered as the extensive sensor network. To simulate the coarser sensor network, 20 by 20 and 10 by 10 meshes are generated and equivalent measurements are taken form the 40 by 40 mesh analysis.

5.2.3 Effect of the Noise in Measurements.—In a real time environment, obtaining ideal (noise free) measurements are very difficult. There is always random amount of noise present in the measured data. Therefore, the detection scheme should be robust enough to withstand certain level of unavoidable noise.

It is very difficult to identify the frequency of noise in the data. According to the literature, white noise and pink noise can cover most of the noise frequency ranges that are typically encountered in many measuring devices used (e.g., accelerometers, strain gauges, etc.). Therefore, white noise and pink noise are considered to study the effects of noise in the damage detection scheme. White noise contains an equal amount of energy per frequency band. For example, the amplitude of the sound will be same from 400 to 500 Hz as it will be from 30100 to 30200 Hz. White noise is considered as bright noise and can be generated by a random number generation function. Pink noise contains equal amount of energy per octave (span of eight notes) band. For example, the amplitude of the sound will be same from 100 to 200 Hz as it is from 200 to 400 Hz or 10000 to 20000 Hz. The ratio of frequency of the highest note to the lowest note in an octave is 2:1. Pink noise can be generated by filtering (averaging) the white noise with a frequency of 8.

In the present study, noise is added to the displacements/mode shapes to simulate the real time environment. To this end, computer routines (written in Visual C++) are used to generate and add the noise. The following equation is used to add the noise.

$$U_i^n\left(orD_i^n\right) = U_i\left(orD_i\right)\left[1 + N_l\left(\frac{\left\|D_i - U_i\right\|}{\left\|U_i(orD_i)\right\|}\right)\right]$$

where U_i^n - undamaged displacements

D_i^n - damaged displacements

N_l - noise level

$\left\|D_i - U_i\right\|$ - Norm of difference of damaged and undamaged displacements

5.1.4 Signal – Feature Reconstruction.—With an eye towards reducing noise intensity and their subsequent adverse effects on the damage detection capability, different types of filters have been developed. The important factor in these so-called "denoising" schemes in the damage detection is retaining the significant, damage-produced features, of the signals. In the last two decades there has been extensive research on the use of wavelet theory. One of the applications of the wavelet theory is denoising. In this research, one dimensional wavelet analysis is used to reduce the noise intensity.

To study the signal feature extraction capability in the global damage detection scheme, noise is added to the data sets. Two types of data sets are considered; i.e., (i) raw displacements, and (ii) processed damage parameter values. The noise level is increased until the damage parameter contour is filled with the noise. The damage parameters with maximum level of noise that can cover the damage are used to extract the feature. The damage parameter values are processed using a one dimensional discrete wavelet analysis to reduce the noise intensity. Daubechies2 (db2) wavelet in the wavelet toolbox, which is in Matlab, is used to reconstruct the feature of the damage parameter. Visual C++ routines are used to interface with Matlab.

5.1.5 False Alarm Test.—Structural response can change due to operational/environmental variability or increase in stiffness. To study the false alarm, two locations on a simply supported plate are considered. At first, both locations are weakened by decreasing the stiffness and then strengthened another time by increasing the stiffness. Damage energy vectors are extracted to distinguish the false alarm case with the actual damage case. Here, the damage vectors will point in the direction of decreased total energy. The vectors pointing outward from a location indicates weakening of the material, whereas, the vectors pointing inward at a location indicates strengthening of the material.

5.1.6 A comparative case study.—J.-C. Hong, et al. (ref. 48) described that wavelet transforms can be adopted for damage detection using the vibration modes of a beam. They also used the wavelet toolbox in Matlab for the simulations. A comparative case study is performed with the simulations described in

reference 48 using the wavelet toolbox in Matlab. In this section of the study, both fine and coarse networks are used to observe the behavior of wavelet analysis in global damage detection.

The fine network is used here for comparison purpose only and also because it is the only case presented in the above reference 48. However, from a practical viewpoint, coarse networks are crucial in judging the effectiveness of any scheme. These coarse network simulations are performed here only using the detection scheme proposed in this report, since unfortunately there were no comparable studies in the available literature searched using other global detection schemes.

Along with these simulation and comparative studies, some actual experiments, which are provided by NASA Glenn Research Center, were also analyzed. These experiments are conducted by Electronic Speckle Pattern Interferometry (ESPI) technique, which is used to determine the correct overall underlying mode shapes and frequencies. Actually, these experimental analyses motivated this study towards investigating the effect of noisy measurements and the feature reconstruction. The results of above cases are presented in the following sections.

5.2 Simulation Results

5.2.1 Low Cyclic Fatigue Failure Mode.—Damage in a structure occurs after a low number of cyclic loadings in Low Cyclic Fatigue failure mode. This type of damage is simulated by a sudden degradation of the Modulus of Elasticity of element/elements in the finite element model. A metallic plate shown in figure 5.1 is considered for the simulations with the damage oriented horizontally.

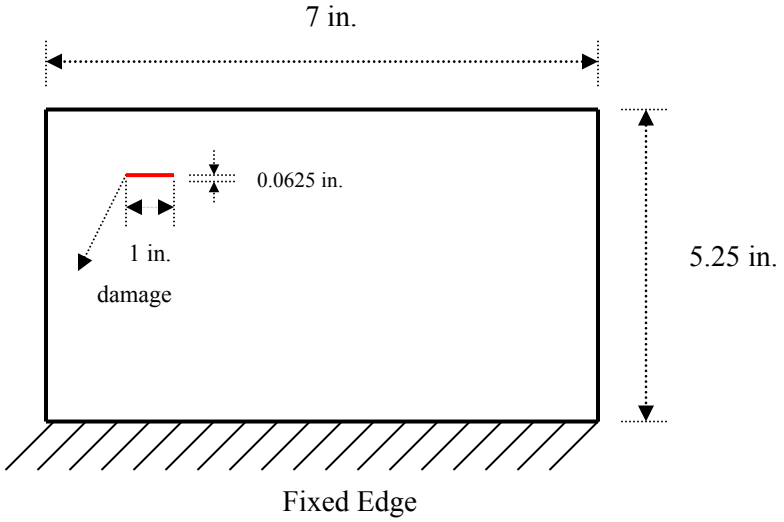


Figure 5.1.—Cantilever Plate with single damage

Material properties:	
Mass of the plate, m:	7.67E-04 lb
Poison's ratio, ν :	0.3
Modulus of elasticity, E:	2.96E+07 ksi
Modulus of elasticity of damaged elements, E_d:	1.00936E+07 ksi
Percentage of material degradation:	66 percent of the actual Modulus
Percentage of damaged area:	0.17 percent of total area
Location of the damage from left top corner:	2.0625 in. in x and 1.5 in. in y – directions

A metallic plate with the above material properties are considered for the simulations. The plate is fixed at the bottom edge and left free at other edges. A mesh of size 40 by 40 is used. A total of six elements are damaged, i.e., the Modulus of Elasticity is reduced to E_d from E, at a location treated as a single damage. Figure 5.2(a) shows the first five mode shapes of the cantilever plate and they can be described as first bending mode, first twisting mode, first mixed (bending and twist) mode, second bending mode, and second twisting mode, respectively. Figure 5.2(b) shows the damage parameter contours in which we can view the location and size of the damage. Figure 5.3 shows the shapes of undamaged, damaged and delta (undamaged – damaged) mode shapes. We can see that the mode shapes of damaged and undamaged are similar, but delta mode shapes are completely different than the actual mode shape of the plate. Figure 5.4 shows the damage energy vectors. The vectors point in the direction of decreased total energy. In all the mode shapes we can see that the vectors are going away from a particular location. That means, that location is weaker compared to the other portions of the plate.

The undamaged and the damaged frequencies are tabulated in tables 5.1 and 5.2 respectively. When we compare them, there is less than 0.01 percent reduction in the damaged frequencies in first five modes. Here, the frequencies do not differ much with the damage. In other words, we can say that, the damage is very tiny since it does not significantly change the frequencies in the damaged plate. This is also further confirmed if one compare the very small difference in the mode changes of figure 5.3(c), as compared to the rather large magnitudes in the almost-identical overall underlying modes (before and after damages) in figures 5.3(a) and 5.3(b). This is of course of practical significance in that the detection at early stages of small-size defects is always desirable. Furthermore, experimental methods needed for the detection are distinctly more demanding in their level of accuracy compared to those that are merely targeting the overall mode shapes as in the traditional modal testing.

TABLE 5.1.—UNDAMAGED FREQUENCIES (CYC/SEC) OF MATERIAL DEGRADATION – SINGLE DAMAGE

Mode-1	149.2480086
Mode-2	295.1841725
Mode-3	722.7485231
Mode-4	942.1421981
Mode-5	1157.100669

TABLE 5.2.—DAMAGED FREQUENCIES (CYC/SEC) OF MATERIAL DEGRADATION – SINGLE DAMAGE

Mode-1	149.2214258
Mode-2	294.7049912
Mode-3	721.222923
Mode-4	939.1733984
Mode-5	1155.178122

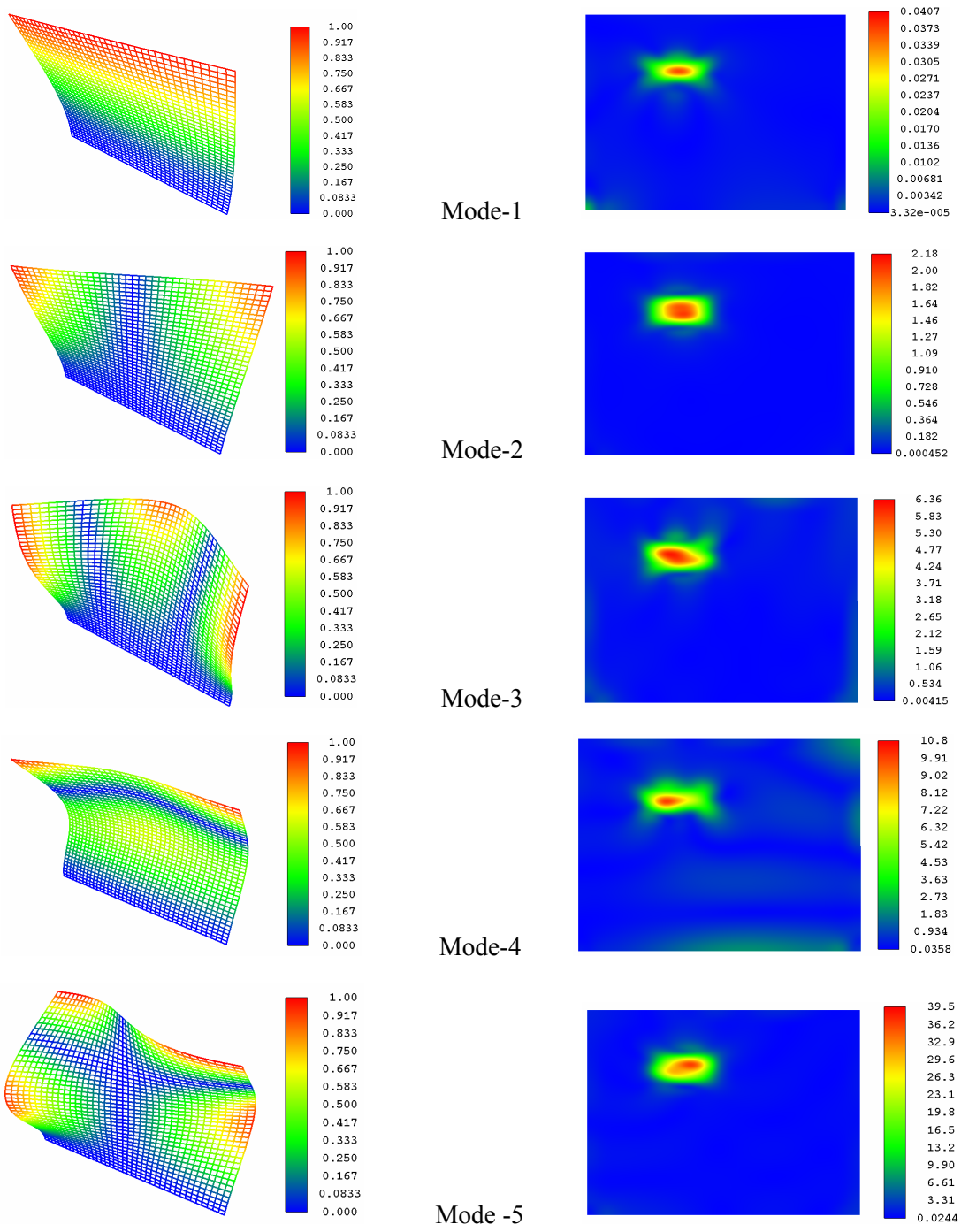


Figure 5.2.—Metallic Plate with Single Damage (a) Mode Shapes (b) Damage Parameter

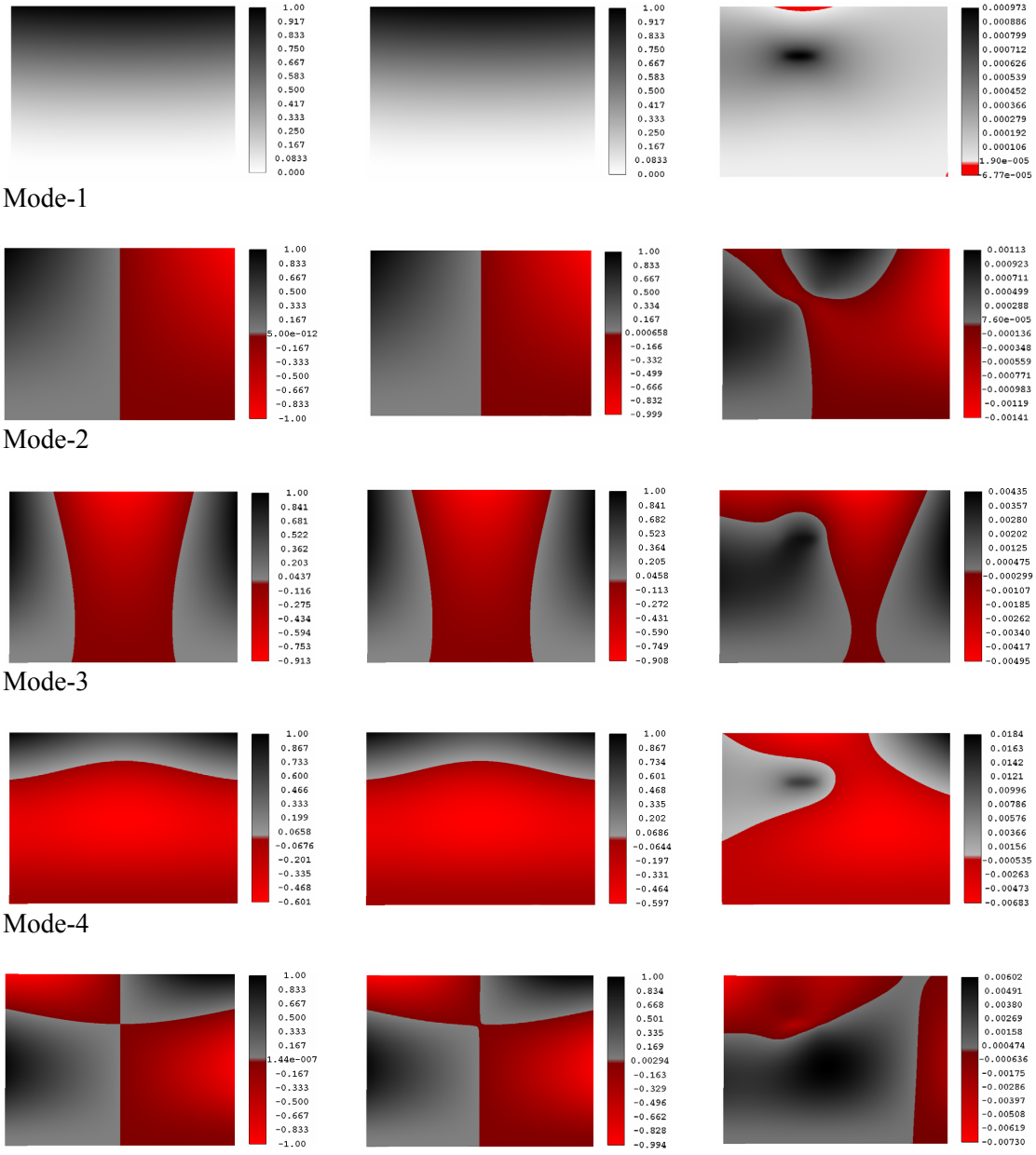


Figure 5.3.—Metallic plate with single damage (a) Undamaged mode shape (b) Damaged mode shape (c) Delta mode shape (undamaged shape–damaged mode shape).

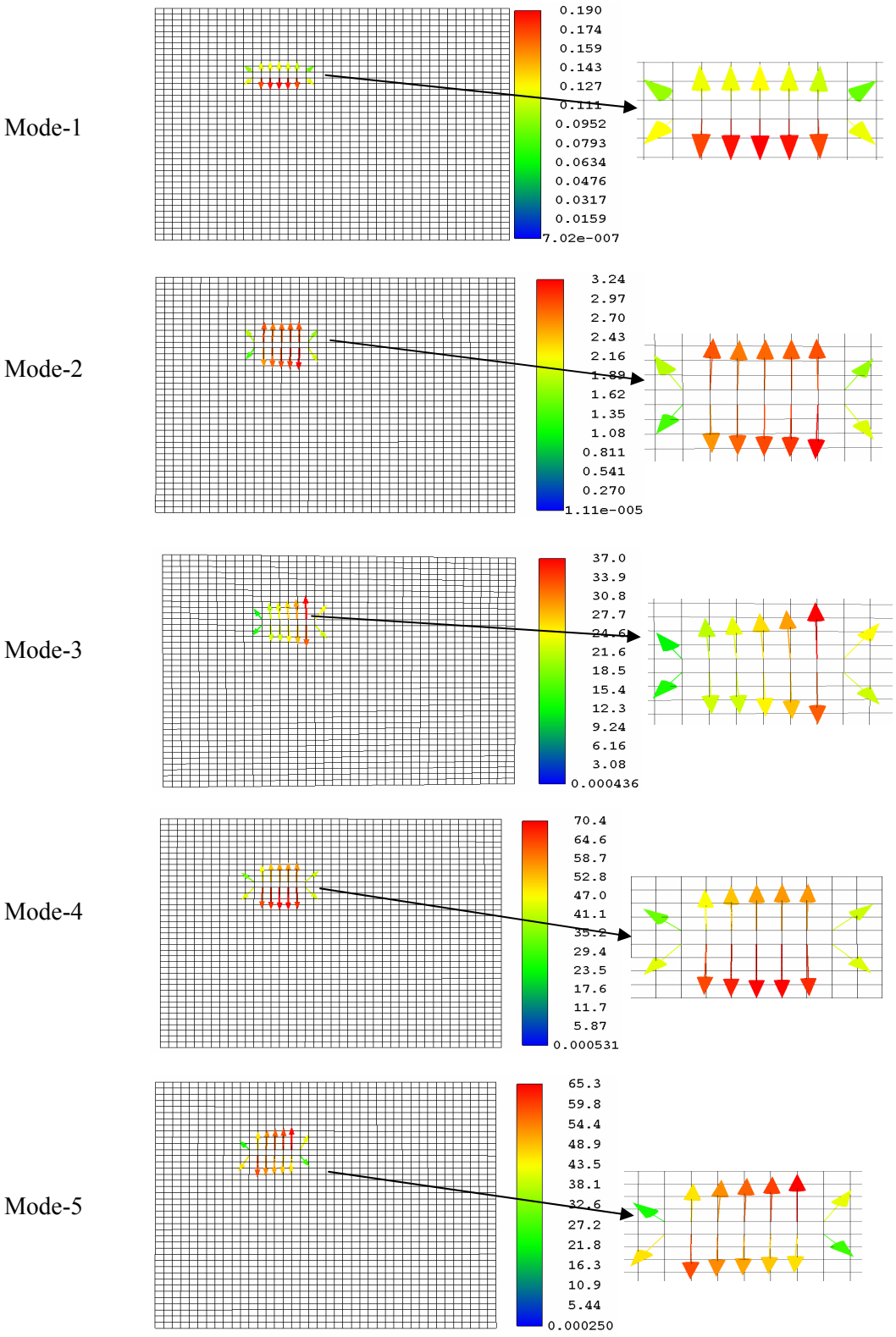


Figure 5.4.—Metallic Plate with Single Damage – Damage nodal force vectors

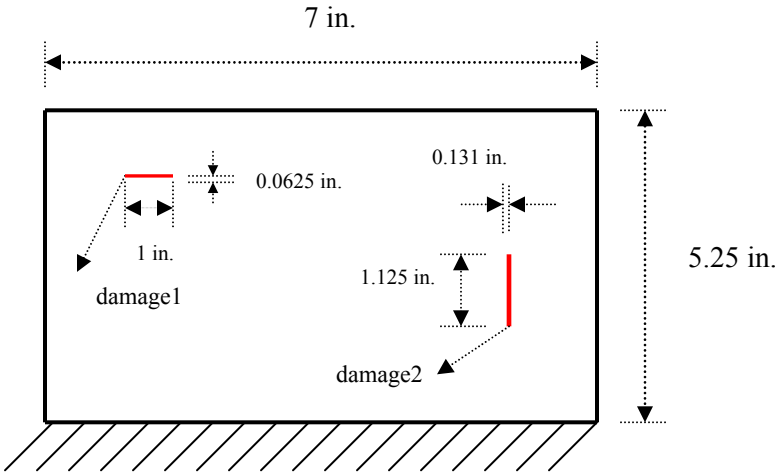


Figure 5.5.—Cantilever plate with multiple damages.

In the following case, two damages are simulated as in figure 5.5. The material properties and the dimensions for damage1 have taken from the previous case.

For the damage2:
Modulus of elasticity of damaged elements for damage2, E_{d2}: 1.04488E+07 ksi
Percentage of material degradation: 65 percent of the actual Modulus
Percentage of damaged area: 0.40 percent of total area
Location of the damage2 from left top corner: 5.075 in. in *x* and 3.625 in. in *y*-directions

Here, seven elements are damaged, i.e., Modulus of Elasticity is reduced to E_{d2} from E, at the location of damage2. Figures 5.6(b) and 5.8 show the damage parameter contours and damage vectors of first five modes respectively. The damage can be identified by observing the first five modes. Here, we can not locate both damages in all five modes, but we can detect the damage if one can provide the first few fundamental modes. Therefore, we can conclude that we need at least a few modes to confirm the existence and location of the damage. Also we can observe that delta mode shapes in figure 5.7(c) are completely different compared to the single damage case. When we compare the undamaged and the damaged frequencies in tables 5.3 and 5.4, damaged frequencies are reduced by less than 0.01 percent in first five modes. The same behavior was observed in single damage case also. As in the single damage case, the difference in the mode changes is very small. Therefore, we can conclude that a tiny amount of damages in structure will not change much in the frequencies and mode shapes, and the level of accuracy needed in experimental methods in the detection is more demanding issue.

TABLE 5.3.—UNDAMAGED FREQUENCIES (CYC/SEC) OF MATERIAL DEGRADATION – MULTIPLE DAMAGES

Mode-1	149.2480086
Mode-2	295.1841725
Mode-3	722.7485231
Mode-4	942.1421981
Mode-5	1157.100669

TABLE 5.4.—DAMAGED FREQUENCIES (CYC/SEC) OF MATERIAL DEGRADATION – MULTIPLE DAMAGES

Mode-1	148.9227814
Mode-2	294.2309847
Mode-3	720.2086541
Mode-4	937.8735199
Mode-5	1154.13901

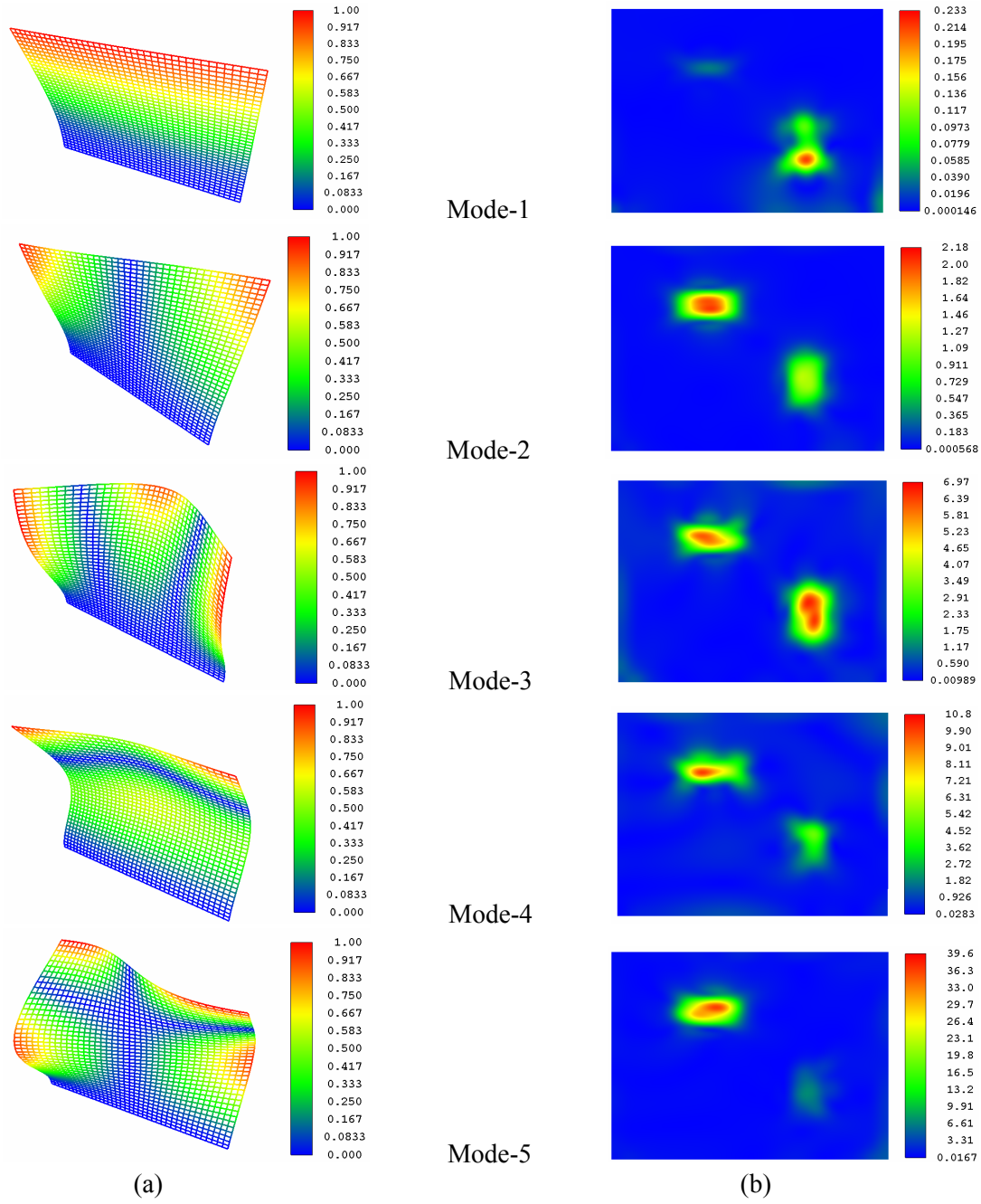


Figure 5.6.—Metallic plate with two damages (a) Mode shapes (b) Damage parameter.

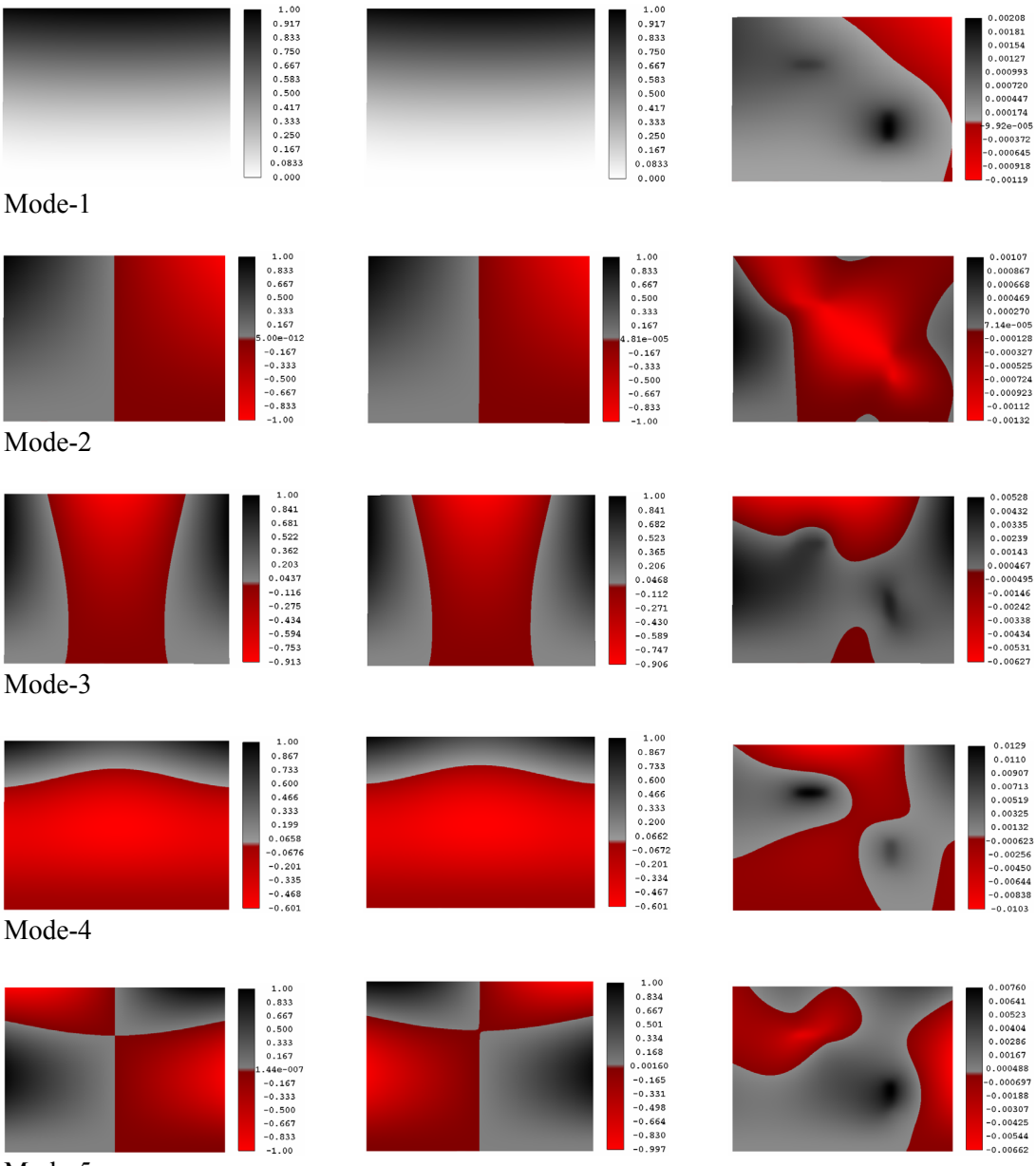


Figure 5.7.—Metallic plate with two damages (a) Undamaged mode shape (b) Damaged mode shape (c) Delta mode shape (undamaged shape–damaged mode shape).

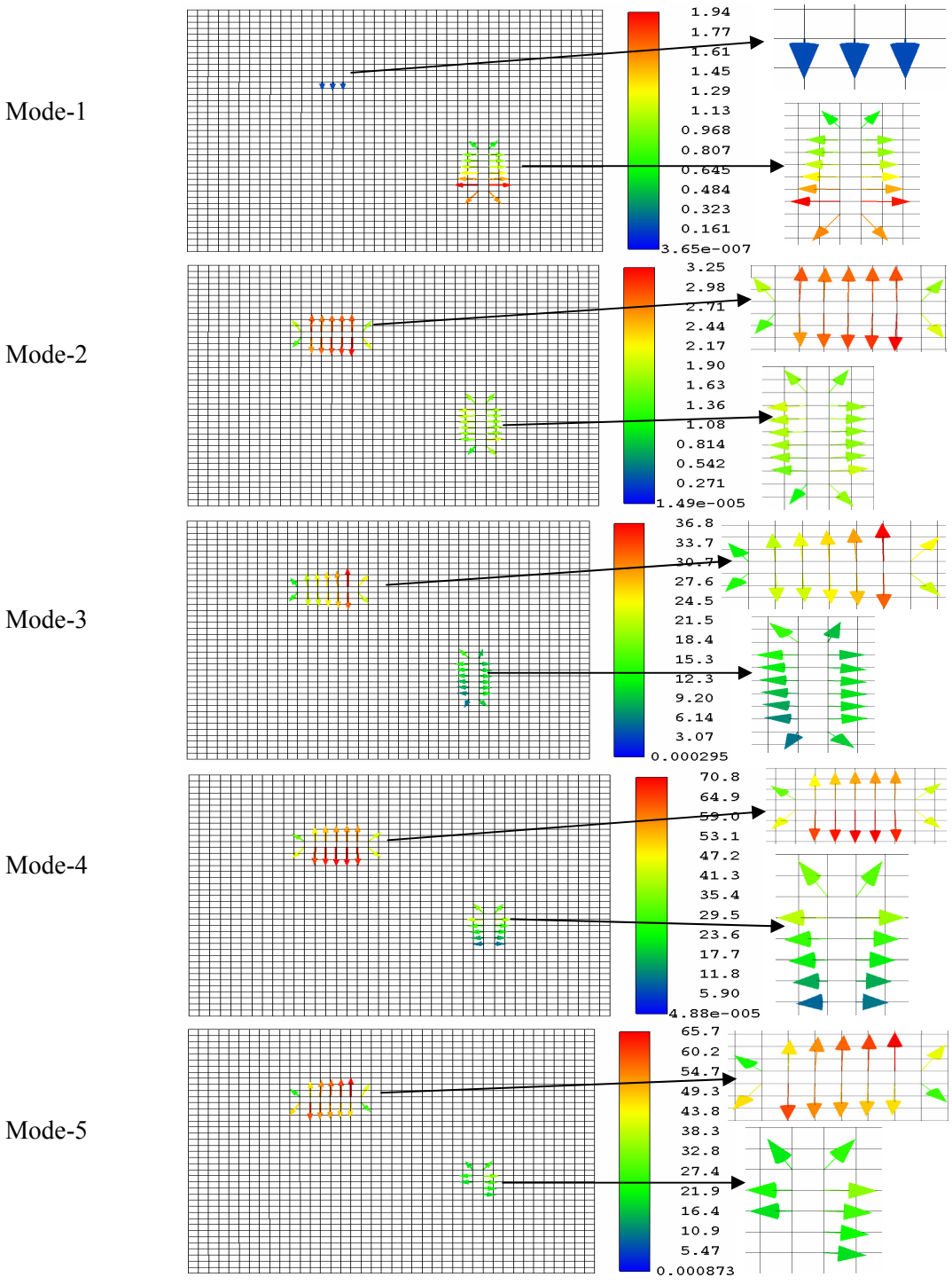


Figure 5.8.—Metallic plate with two damages – damage nodal force vectors.

5.2.2 High Cyclic Fatigue Failure Mode.—The damage in a structure can occurs after a high number of cyclic loadings in high Cyclic Fatigue failure mode. This type of damage/line crack is simulated by creating multiple nodes at the same geometric location in the finite element model. A metallic plate in figure 5.9 is considered for simulations. The plate material properties are taken same as previous cases.

A crack of length 175 mil (1 mil = 0.001 in.), oriented horizontally, on a plate fixed at one edge is simulated with a mesh of 20 by 10 size. Initially a mesh of 80 by 20 size is generated and an extra node is created at the location of damage. Nodal connectivity is changed using the extra node to simulate the crack. Then the equivalent nodal displacements on a mesh of 20 by 10 size are considered to extract the first five fundamental modes. The results are presented from figures 5.10 to 5.12. In figure 5.12, damage vectors of mode 5 are shown at two locations unlike other modes. This is happened due to the false alarm, which is caused by the processing of huge data files. But, overall we can confirm the existence and location of the damage by considering the first five fundamental modes. Therefore, to confirm the existence and the location of the damage we require at least few mode shapes. The table 5.5 shows the undamaged frequencies of the first five modes.

TABLE 5.5.—UNDAMAGED FREQUENCIES (CYC/SEC) OF LINE CRACK

Mode-1	149.1728511
Mode-2	294.822034
Mode-3	721.1457221
Mode-4	943.1049457
Mode-5	1158.669922

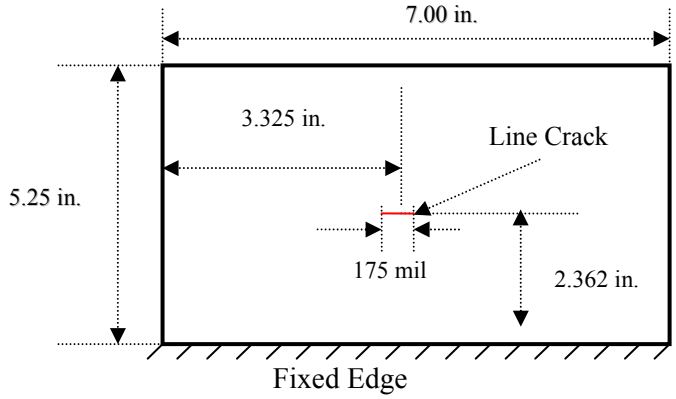
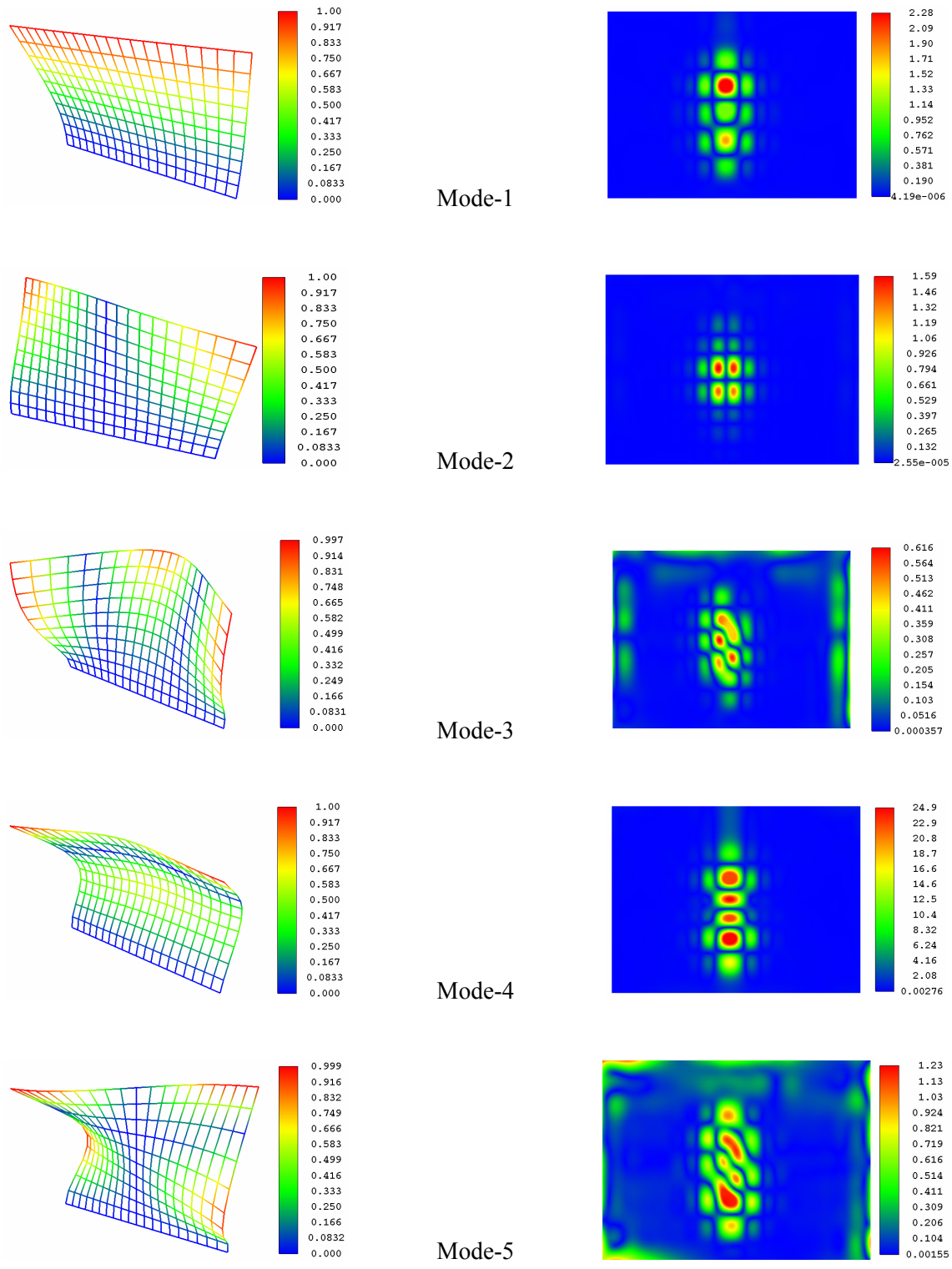


Figure 5.9.—Cantilever plate with line crack.



(a) (b)

Figure 5.10.—Metallic plate with a line crack (a) Mode shapes (b) Damage parameter.

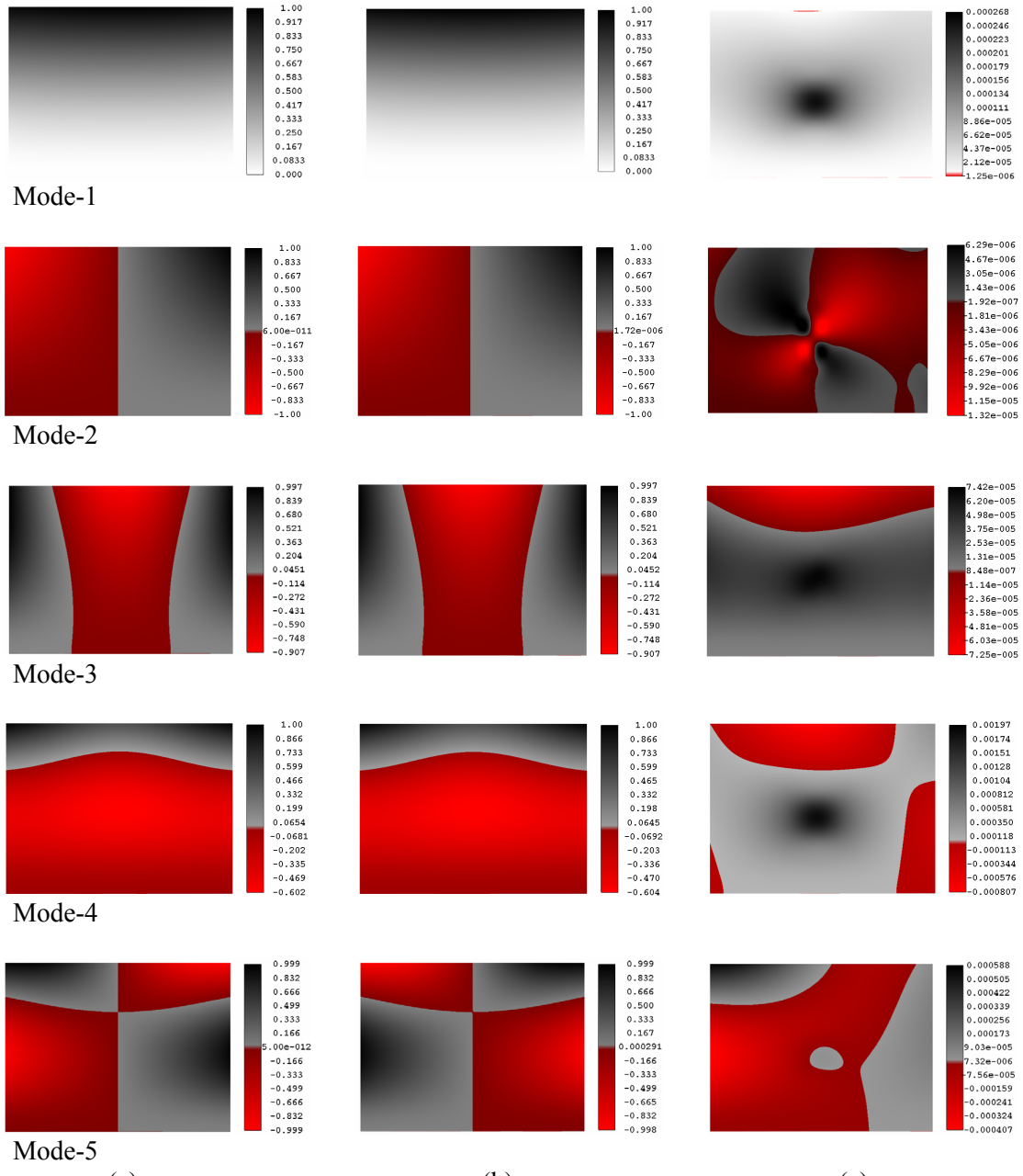
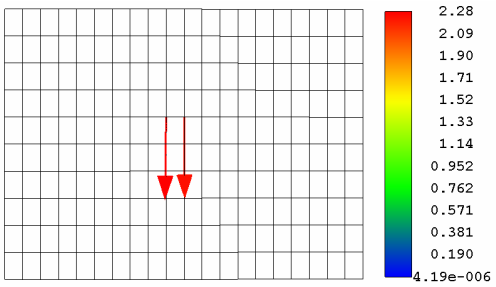
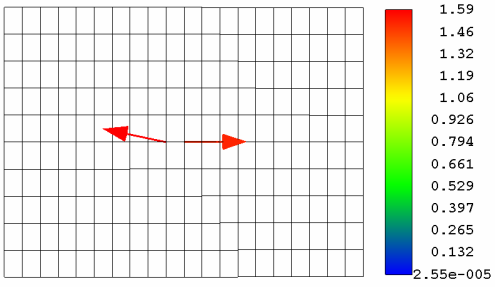


Figure 5.11.—Metallic plate with a line crack (a) Undamaged mode shape (b) Damaged mode shape (c) Delta mode shape.

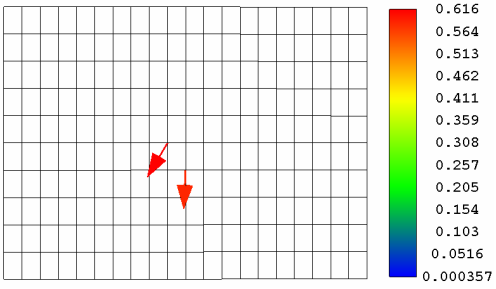
Mode-1



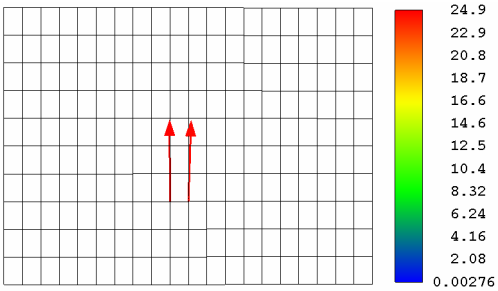
Mode-2



Mode-3



Mode-4



Mode-5

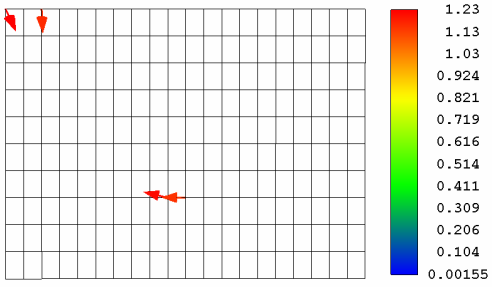


Figure 5.12.—Metallic plate with a line crack – damage nodal force vectors.

5.2.3 Corrosion in the Structure.— Corrosion of the material is also a type of damage to the structure. When corrosion of the structure begins, we can observe tiny holes. This is simulated by eroding an element from the mesh geometry. Two different mesh sizes, 40 by 40 and 120 by 120, are considered for this case. The plate in figure 5.13 is considered with the material properties as in previous cases.

A finite element model is created using ABAQUS with different mesh sizes and an element is deleted from the geometry to simulate the corrosion effect. The first five fundamental modes are generated using frequency analysis and the undamaged frequencies are listed in the tables 5.6 and 5.7. Displacements of each mode are saved and entered into the data input file of the detection scheme for damage detection.

The features presented in the figures from figures 5.14 to 5.19 describe the existence and the location of the damage in both cases. The figures 5.14 and 5.16 show the existence of the damage for 40 by 40 mesh analysis, where 0.0625 percent of total area is damaged and the figures 5.17 and 5.19 show the existence of the damage for 120 by 120 mesh analysis, where 0.0067 percent of total area is damaged. As expected, in both cases, the delta mode shapes, presented in figures 5.15 and 5.18, are completely different than the actual mode shapes.

TABLE 5.6.—UNDAMAGED FREQUENCIES (CYC/SEC)
OF CORROSION – MESH 40 BY 40

Mode-1	144.133321
Mode-2	285.0233531
Mode-3	698.1044529
Mode-4	910.8488979
Mode-5	1118.65208

TABLE 5.7.—UNDAMAGED FREQUENCIES (CYC/SEC)
OF CORROSION – MESH 120 BY 120

Mode-1	149.2513612
Mode-2	295.1571222
Mode-3	722.5704527
Mode-4	941.9485506
Mode-5	1157.162608

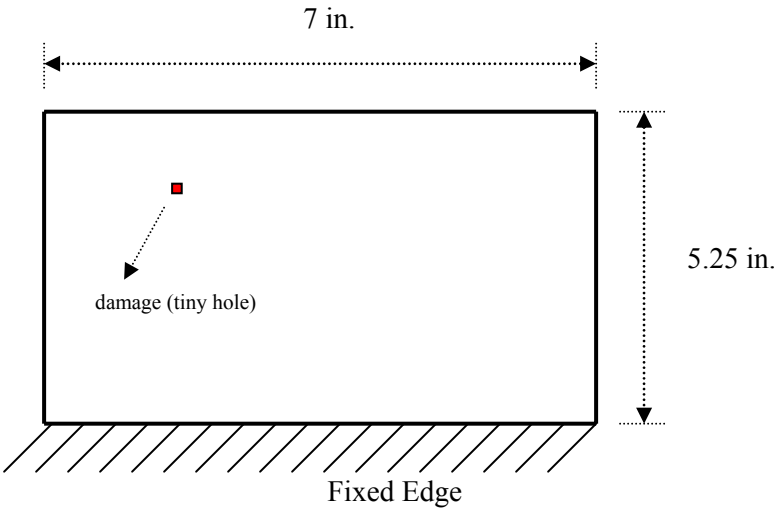


Figure 5.13.—Cantilever Plate with a tiny hole.

Location of the damage from left top corner: 2.0625 in. in x – direction
1.5 in. in y – direction
Percentage of damaged portion (40 by 40 mesh): 0.0625 percent of total area
Percentage of damaged portion (120 by 120 mesh): 0.0067 percent of total area

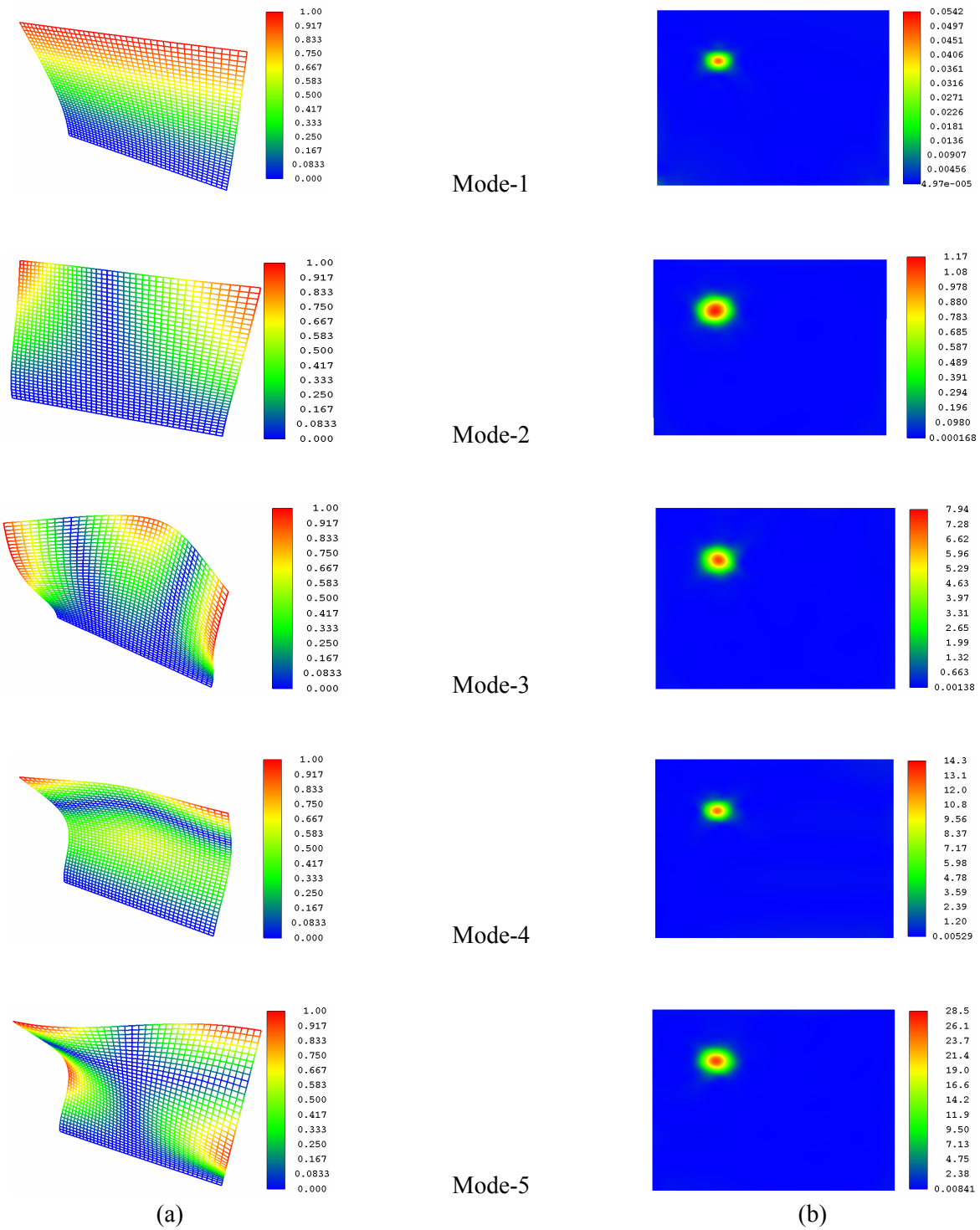


Figure 5.14.—Metallic plate (40 by 40 mesh) with a tiny hole due to corrosion (a) Mode shapes (b) Damage parameter.

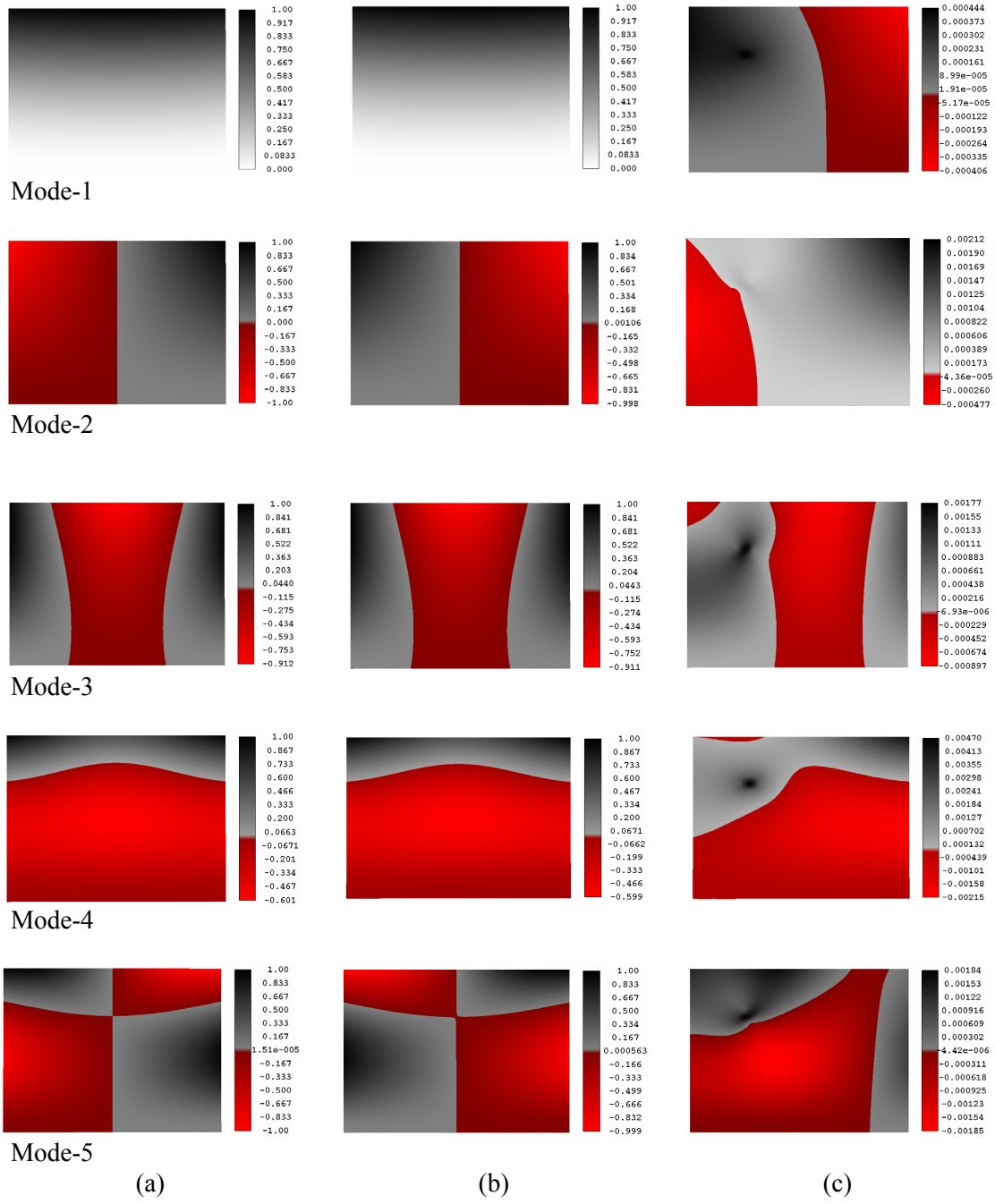


Figure 5.15.—Metallic plate (40 by 40 mesh) with a tiny hole due to corrosion (a) Undamaged mode shape (b) Damaged mode shape (c) Delta mode shape.

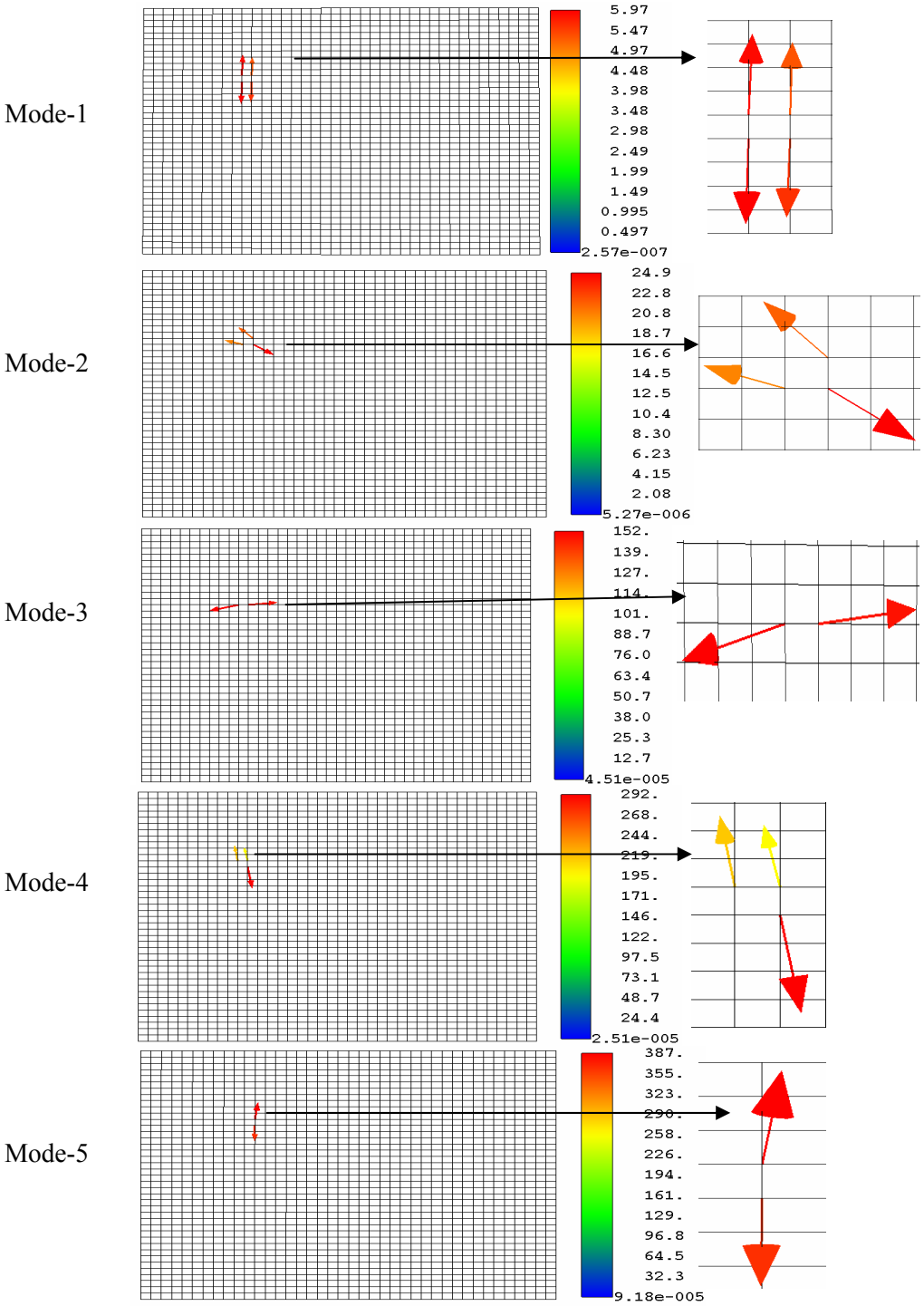


Figure 5.16.—Metallic plate (40 by 40mesh) with a tiny hole-damage nodal force vectors

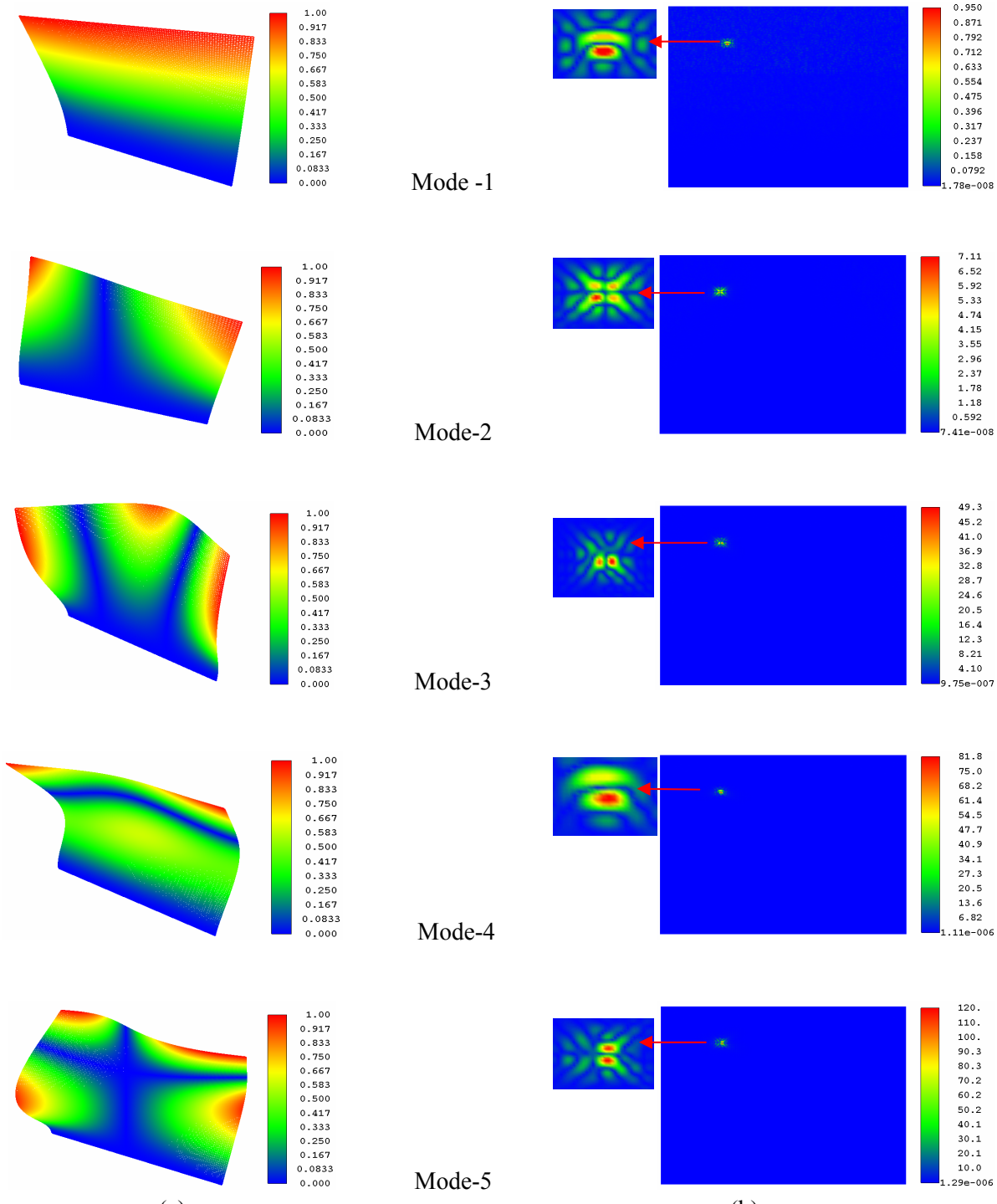


Figure 5.17.—Metallic plate (120 by 120 mesh) with a tiny hole due to corrosion (a) Mode shapes (b) Damage parameter.

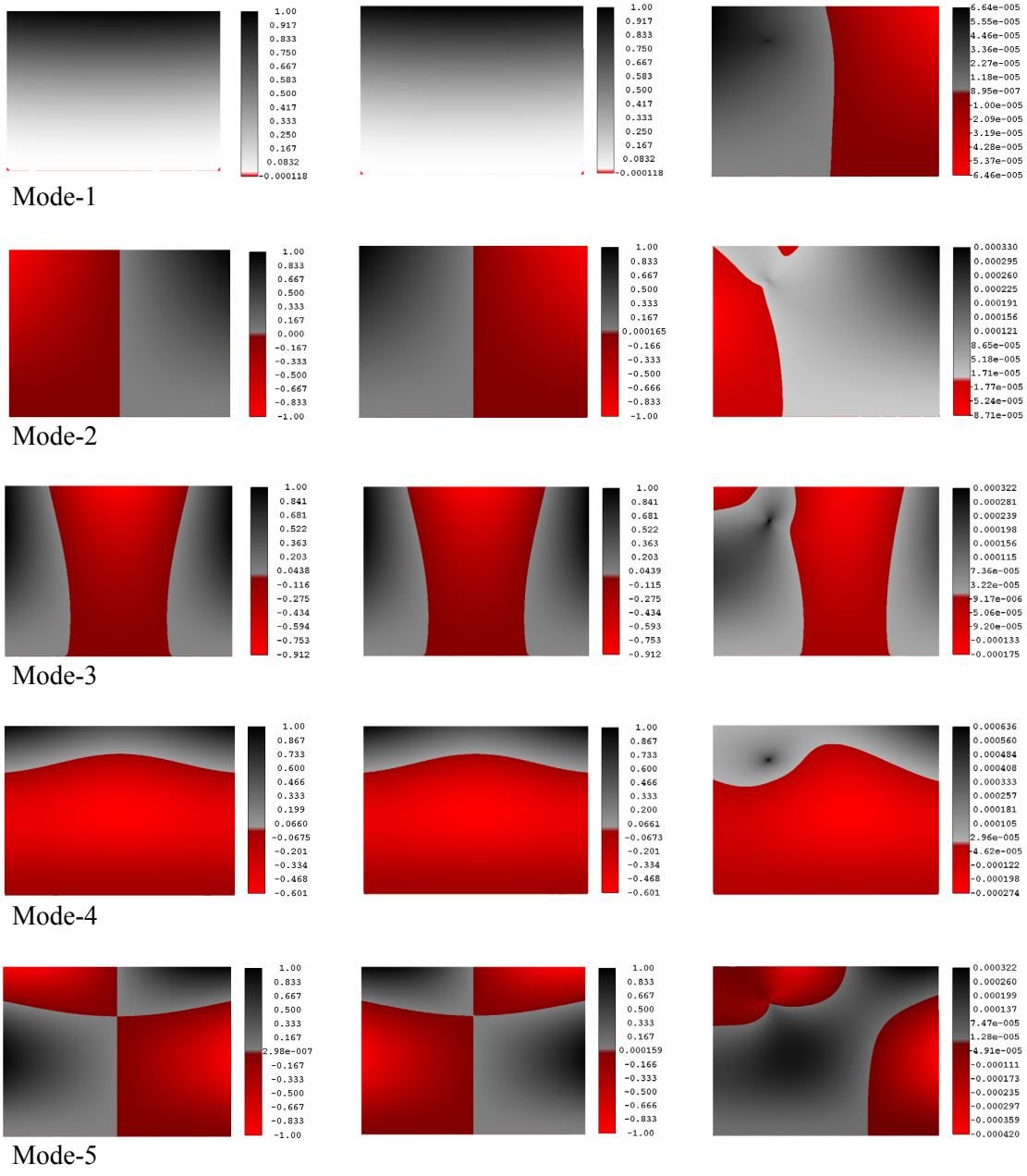


Figure 5.18.—Metallic plate (120 by 120 mesh) with a tiny hole due to corrosion (a) Undamaged mode shape (b) Damaged mode shape (c) Delta mode shape.

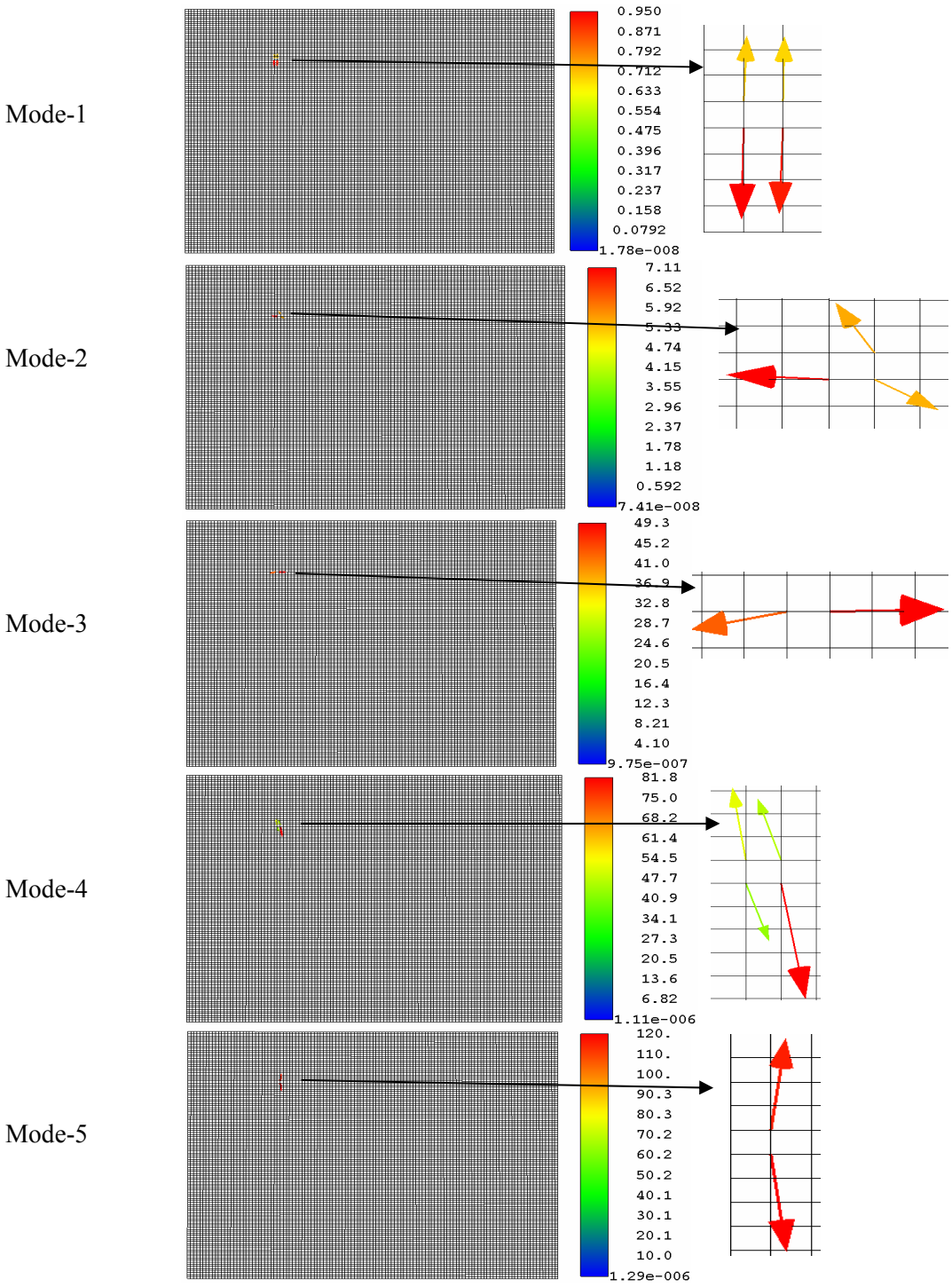


Figure 5.19.—Metallic plate (120 by 120 mesh) with a tiny hole–damage nodal force vectors.

5.2.4 Sensor Network on the Structure in Global Damage Detection.—The effect of the sensor network is a major issue in global damage detection techniques. Therefore, a case study is performed on the density of sensor network. Three sizes of arrangement of sensor networks are considered. For simulations, three mesh sizes are taken viz. extensive network (40 by 40), intermediate network (20 by 20), and coarse network (10 by 10). The case in section 5.2.1 is considered as an extensive network and equivalent displacements of coarser networks are extracted from this extensive network.

Important features are extracted from the simulations using Eview. Mode shapes, delta mode shapes, and damage parameter values are extracted for comparison. In all three cases, mode shapes are the same as in cantilever plate. As expected, delta mode shapes are completely different than the actual mode shapes. We are able to extract the feature of the damage for every size of the network arrangement. If we observe the results carefully, the size of the damage is widened from fine to coarse network. Since this is a finite element analysis, extraction of size of the damage is proportional to the size of the elements in the mesh.

The proposed detection scheme in this report is robust in the extraction of the damage even with coarser sensor networks. The required figures are presented from the figures 5.20 to 5.22.

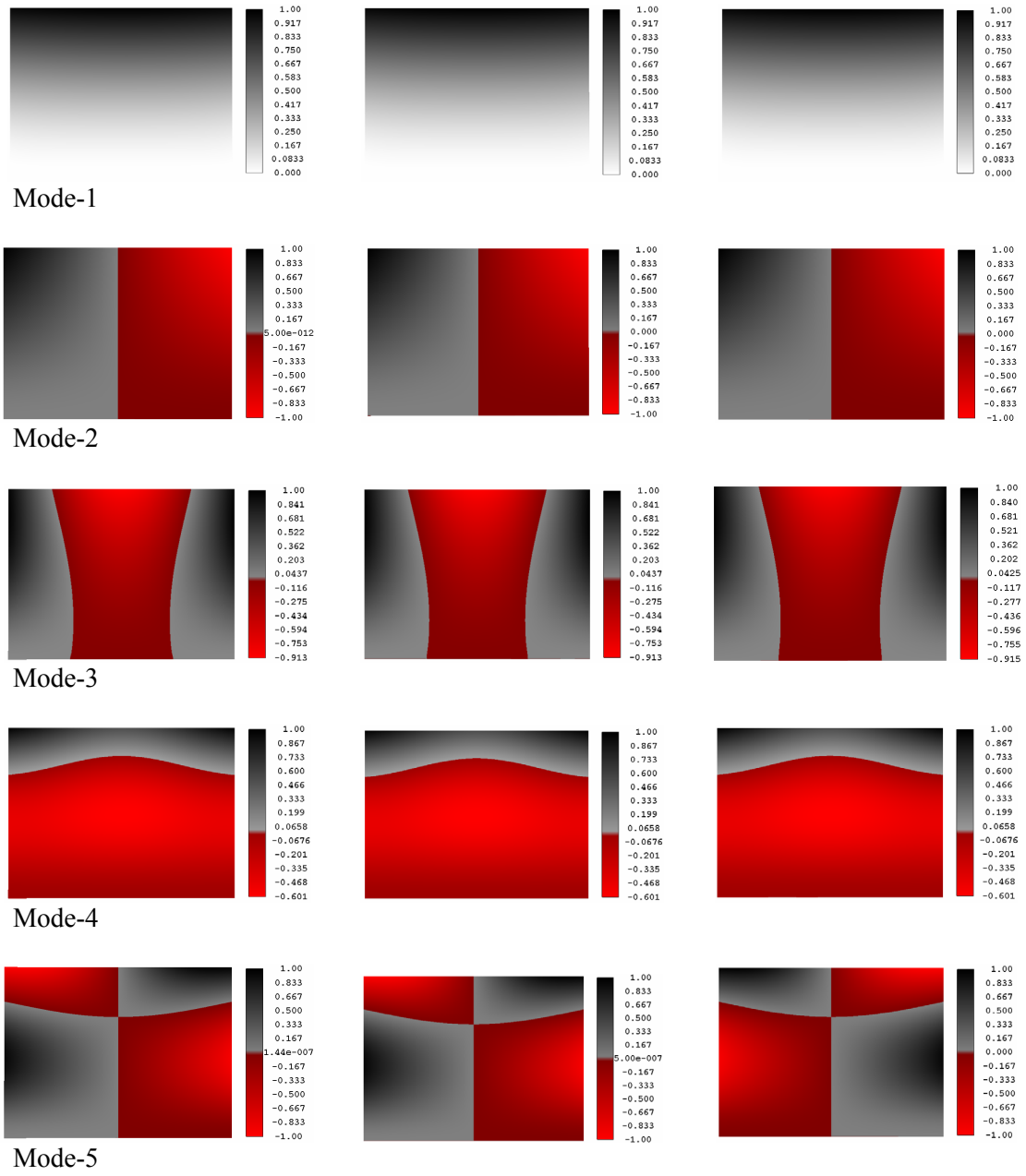


Figure 5.20.—Mode shapes (a) Extensive sensor network
(b) Intermediate sensor network (c) Coarse sensor network.

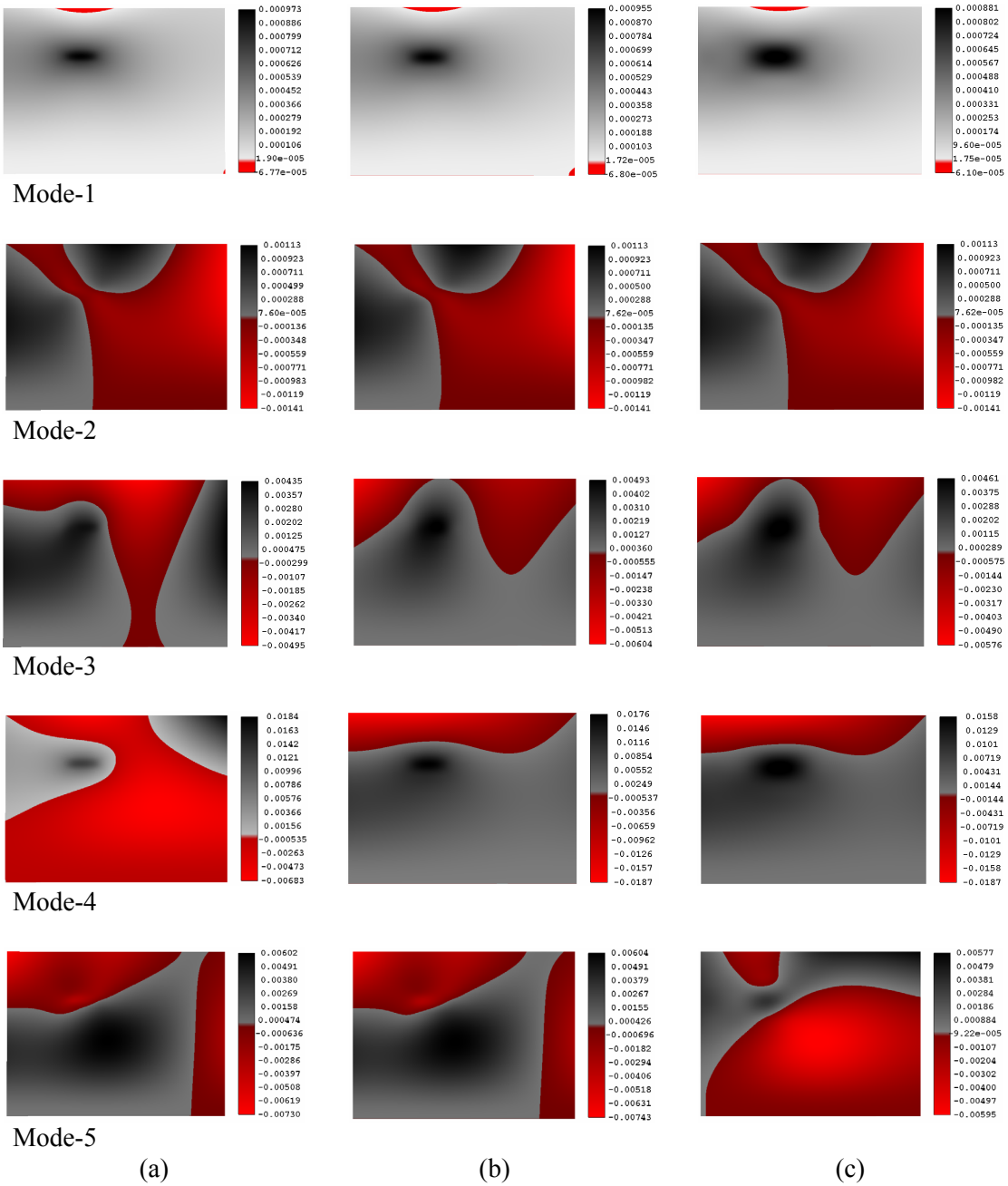


Figure 5.21.—Delta mode shapes (a) Extensive sensor network
(b) Intermediate sensor network (c) Coarse sensor network.

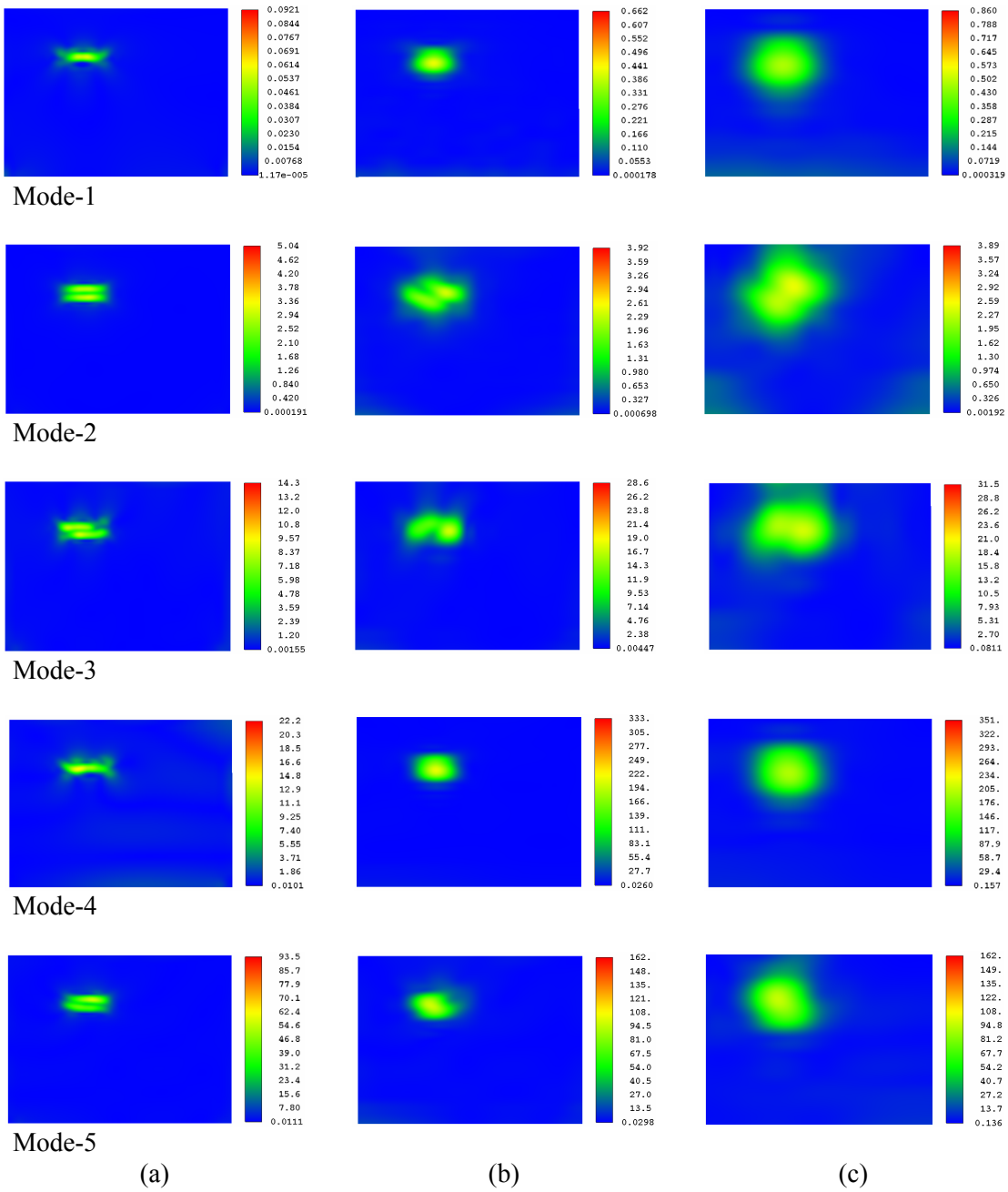


Figure 5.22.—Damage parameter (a) Extensive sensor network
(b) Intermediate sensor network (c) Coarse sensor network.

5.2.5 Analysis of Experimental Files.—NASA Glenn Research Center is using Electronic Speckle Pattern Interferometry (ESPI) experimental setup for the measurement of overall underlying mode shapes, which is used here for the Global Damage Detection. Experimental files provided by NASA are analyzed to detect the presence of the damage. When we observe the undamaged, damaged, and delta mode shapes, the delta mode shapes have similar pattern to that of the actual mode shapes which is unlike to the previous cases. The damage parameter also shows the same pattern, instead of showing sharp damage.

The undamaged frequencies are presented in table 5.8. The figures 5.23 and 5.24 show the mode shapes and damage parameter contours respectively.

TABLE 5.8.—UNDAMAGED FREQUENCIES (CYC/SEC)
OF EXPERIMENTAL RESULTS.

Mode-1	134.600000
Mode-2	274.700000
Mode-3	702.100000
Mode-5	1070.500000

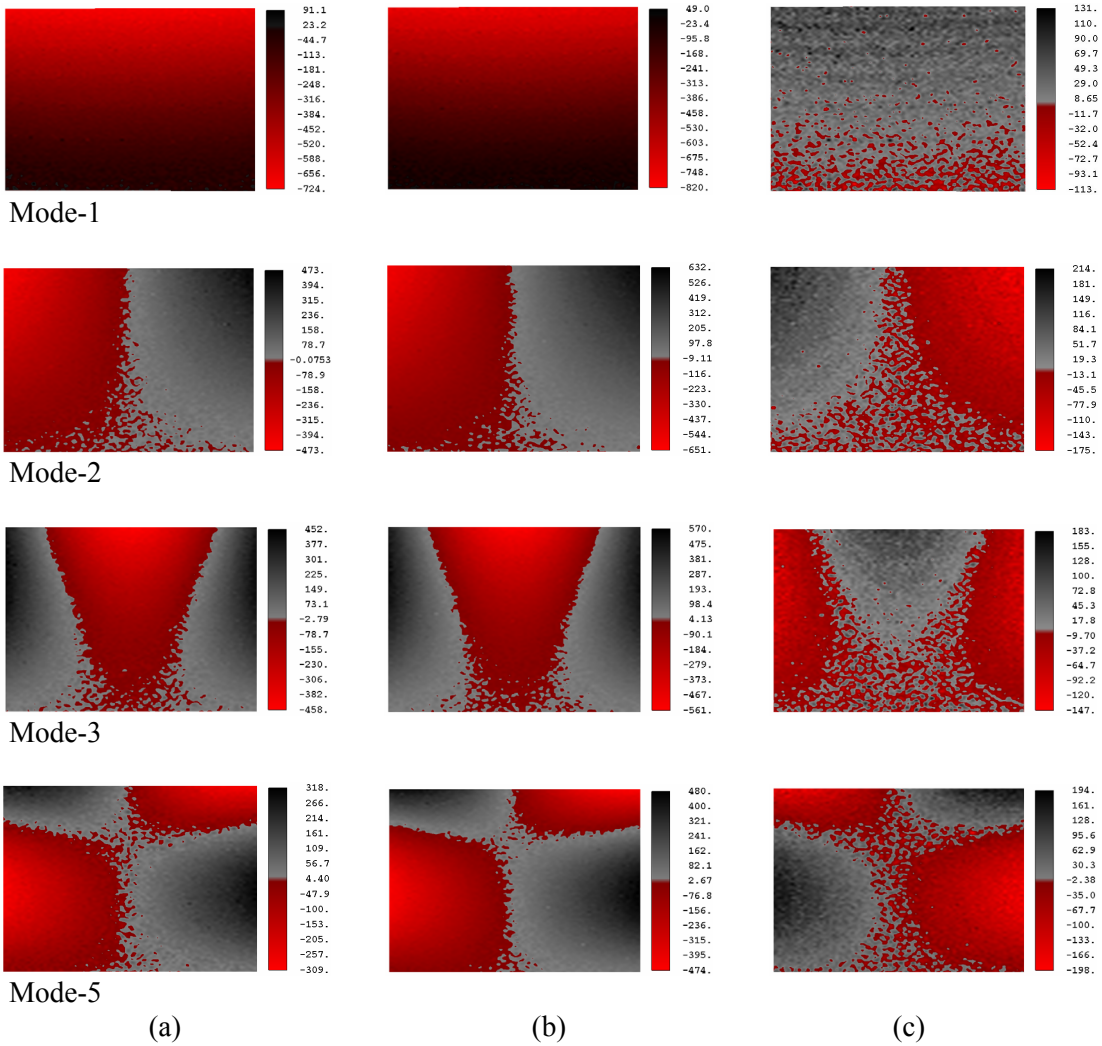


Figure 5.23.—Experimental results with single damage (a) Undamaged mode shape (b) Damaged mode shape (c) Delta mode shape.

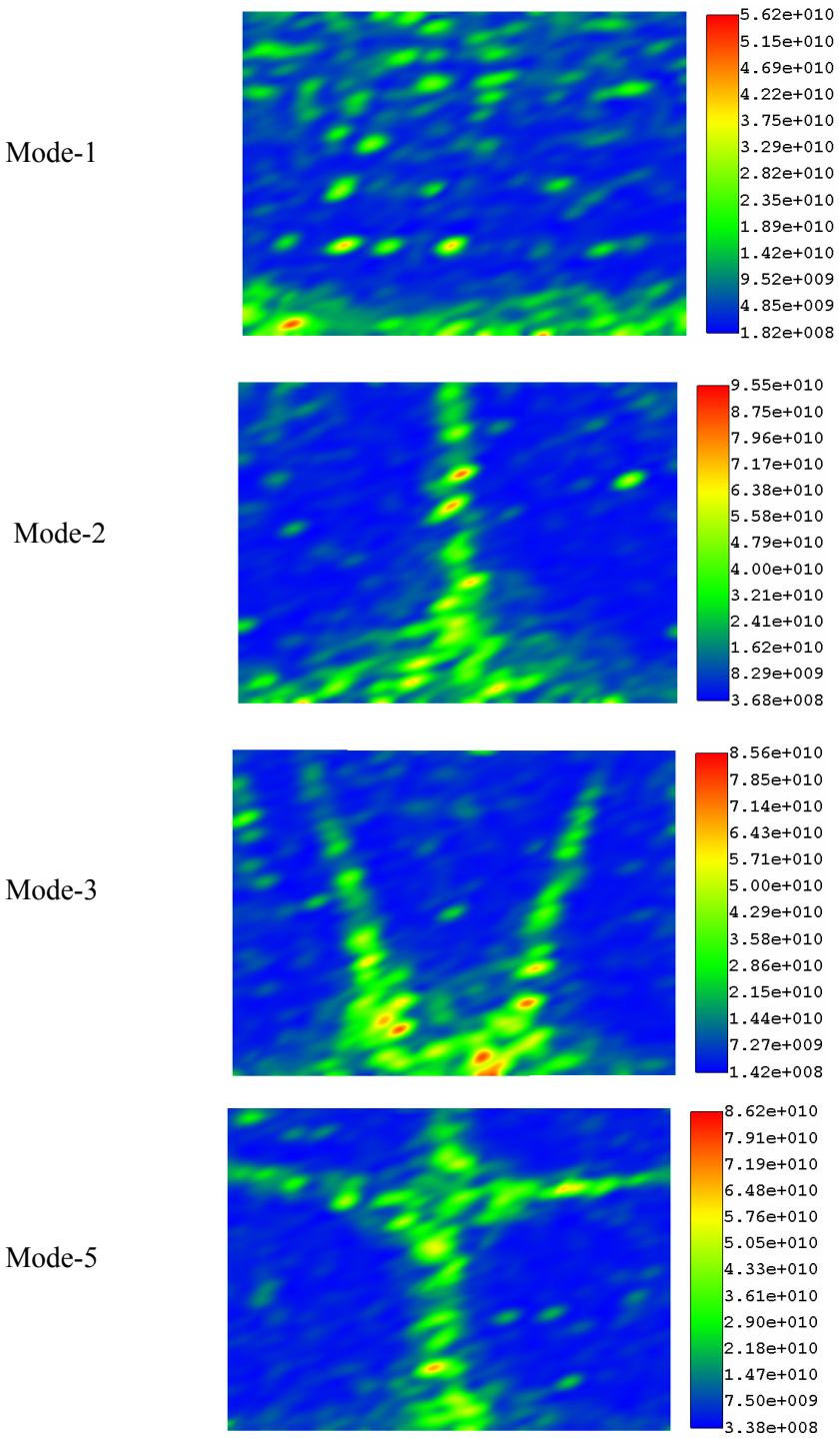
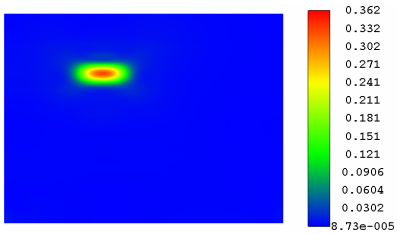


Figure 5.24.—Experimental results with single damage – damage parameter.

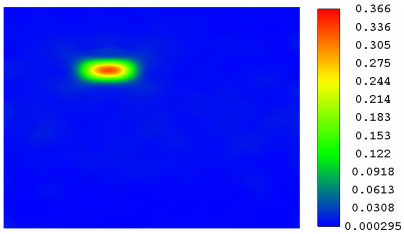
5.2.6 Effect of the Noise in Measurements.—To study the effect of the noise in the detection scheme, white noise and pink noise are added to first five fundamental mode shapes. The damage parameter is shown to be able to extract the damage. In addition, each mode is able to withstand a certain level of noise. The ideal (noise free) case is considered from section 5.2.1.

From figures 5.25 to 5.29 shows the damage parameter contours for fist five modes. In particular, figure 5.25(b) shows the damage parameter of mode-1 with white noise as vectors, where we can increase

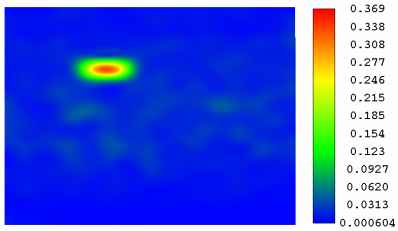
the vector threshold to retain those vectors with high magnitudes. Each figure shows the damage parameter contours of a mode with various levels of noise, which include both white and pink noises. These figures emphasize that each mode have different critical levels of noise. The level of noise that each mode can withstand is summarized in table 5.1.



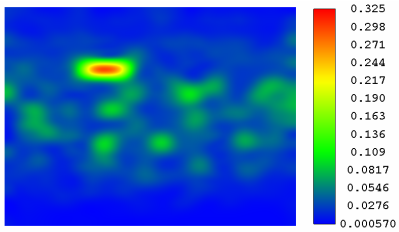
Mode-1 with 0% white noise



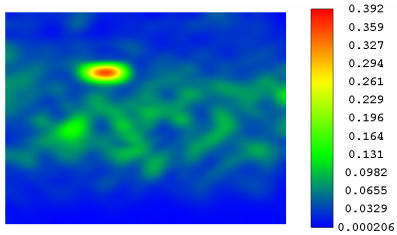
Mode-1 with 10% white noise



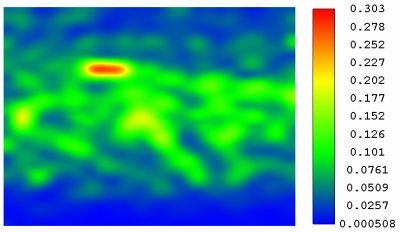
Mode-1 with 35% white noise



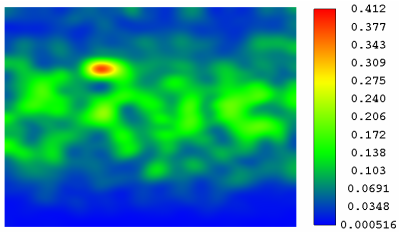
Mode-1 with 70% white noise



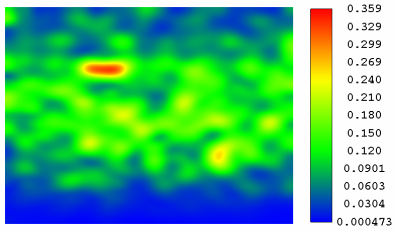
Mode-1 with 105% white noise



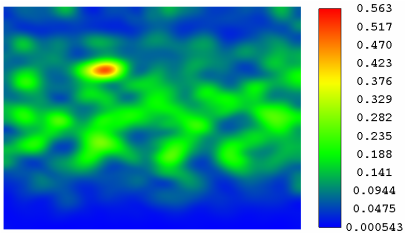
Mode-1 with 140% white noise



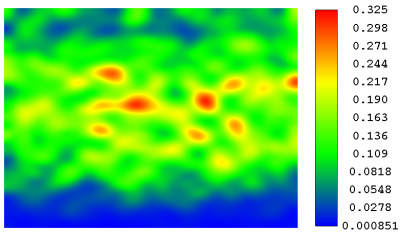
Mode-1 with 175% white noise



Mode-1 with 210% white noise

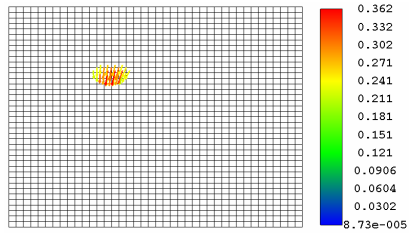


Mode-1 with 245% white noise

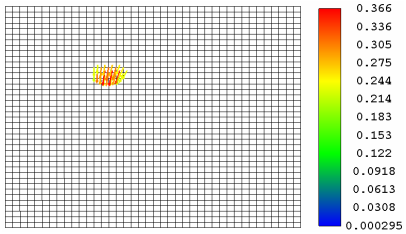


Mode-1 with 280% white noise

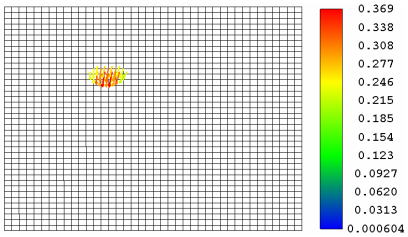
Figure 5.25.—Mode-1 with white noise (a) damage parameter contour view.



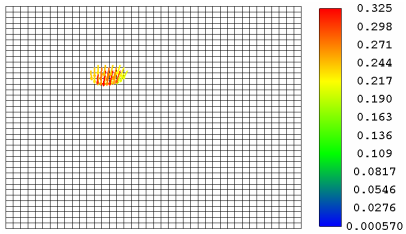
Mode-1 with 0% white noise



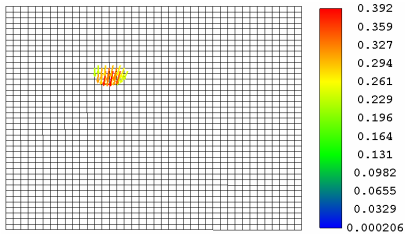
Mode-1 with 10% white noise



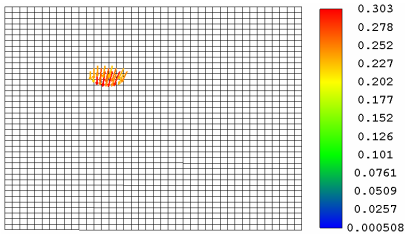
Mode-1 with 35% white noise



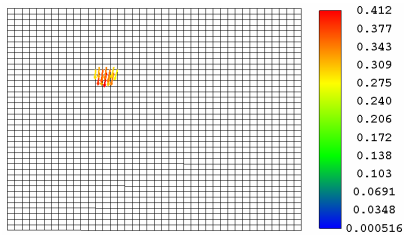
Mode-1 with 70% white noise



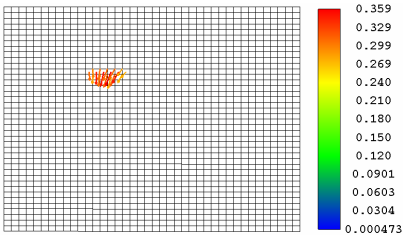
Mode-1 with 105% white noise



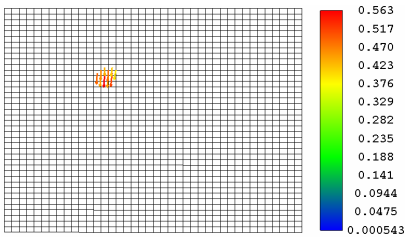
Mode-1 with 140% white noise



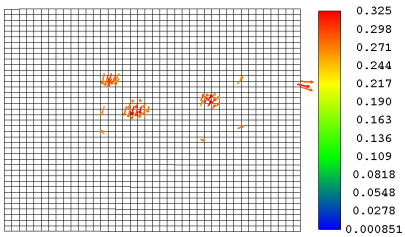
Mode-1 with 175% white noise



Mode-1 with 210% white noise

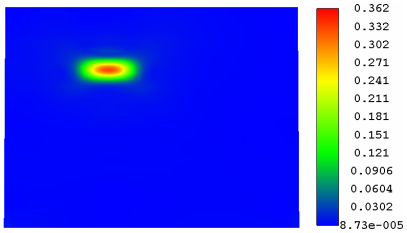


Mode-1 with 245% white noise

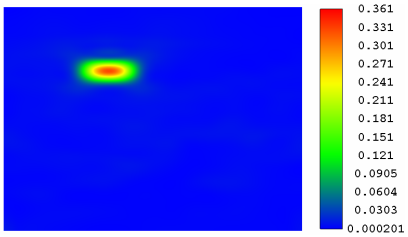


Mode-1 with 280% white noise

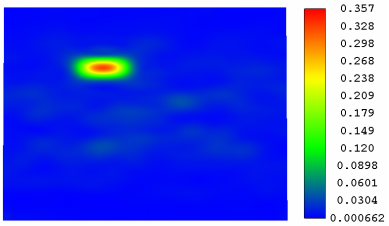
Figure 5.25.—Mode-1 with white noise (b) damage parameter vectors view.



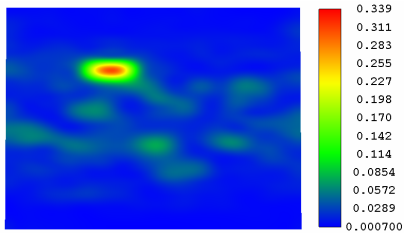
Mode-1 with 0% pink noise



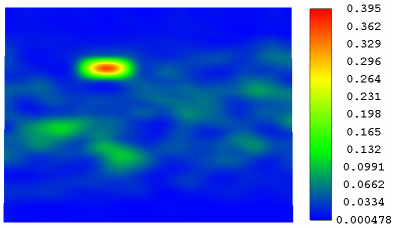
Mode-1 with 10% pink noise



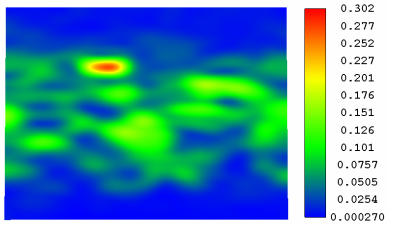
Mode-1 with 35% pink noise



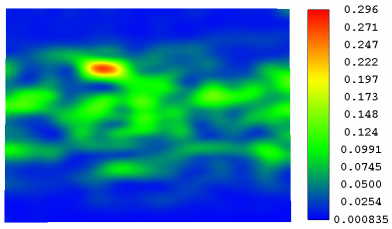
Mode-1 with 70% pink noise



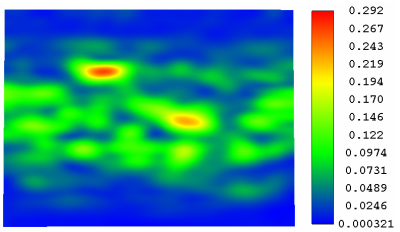
Mode-1 with 105% pink noise



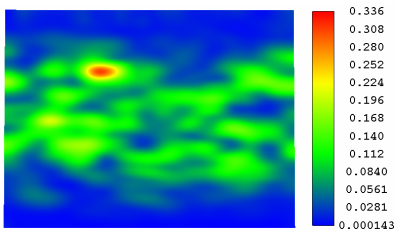
Mode-1 with 140% pink noise



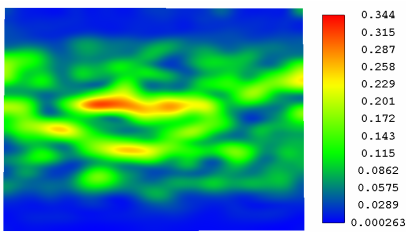
Mode-1 with 175% pink noise



Mode-1 with 210% pink noise

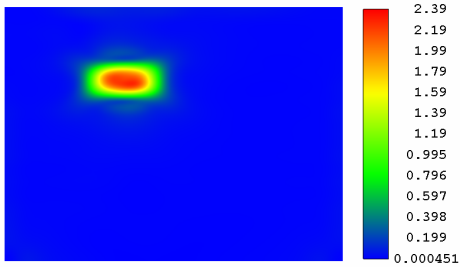


Mode-1 with 245% pink noise

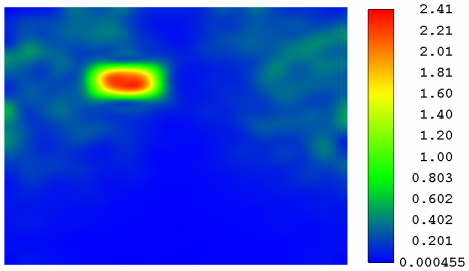


Mode-1 with 280% pink noise

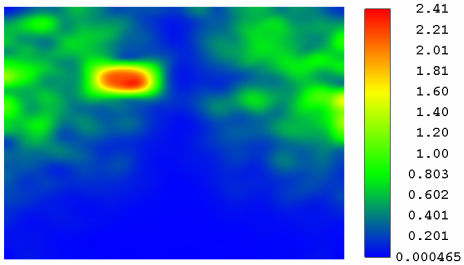
Figure 5.26.—Mode-1 with pink noise



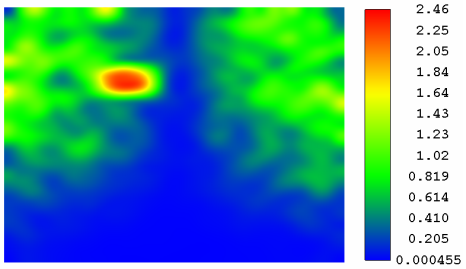
Mode-2 with 0% white noise



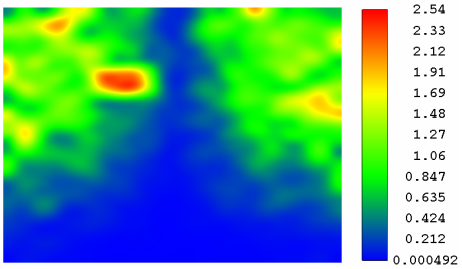
Mode-2 with 35% white noise



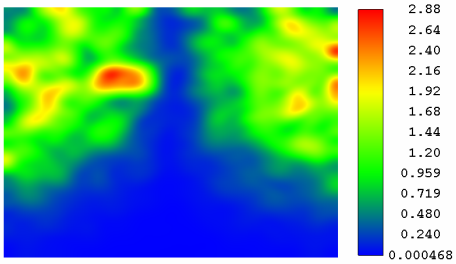
Mode-2 with 70% white noise



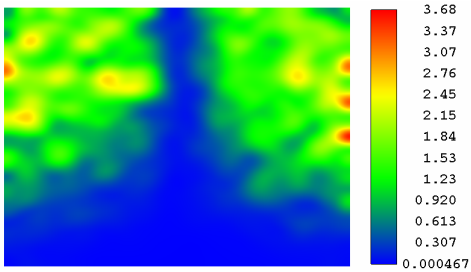
Mode-2 with 105% white noise



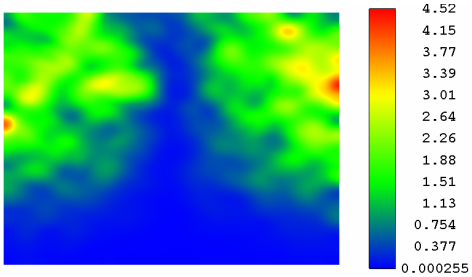
Mode-2 with 140% white noise



Mode-2 with 175% white noise

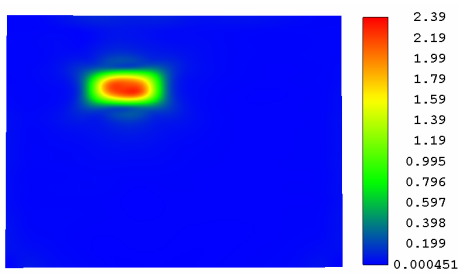


Mode-2 with 210% white noise

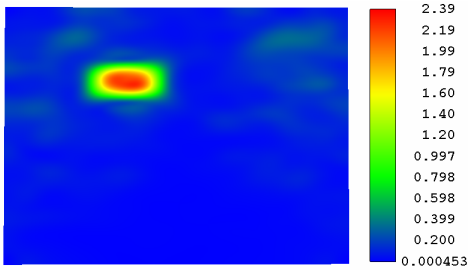


Mode-2 with 245% white noise

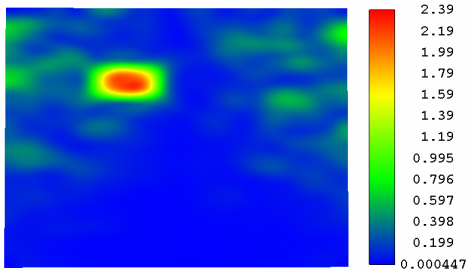
Figure 5.27.—Mode-2 with white noise



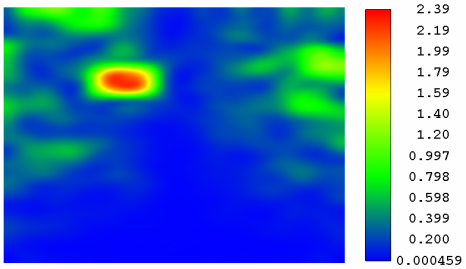
Mode-2 with 0% pink noise



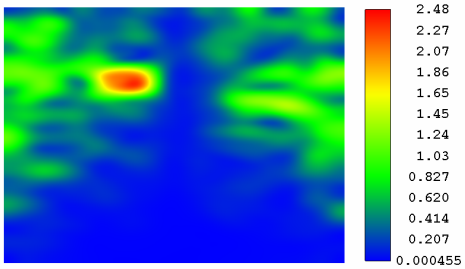
Mode-2 with 35% pink noise



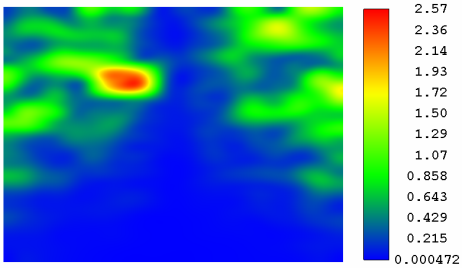
Mode-2 with 70% pink noise



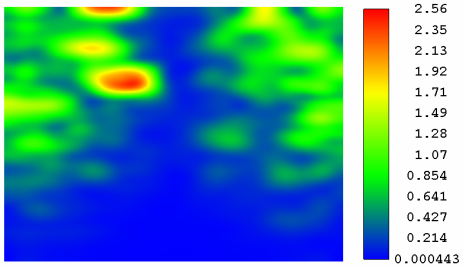
Mode-2 with 105% pink noise



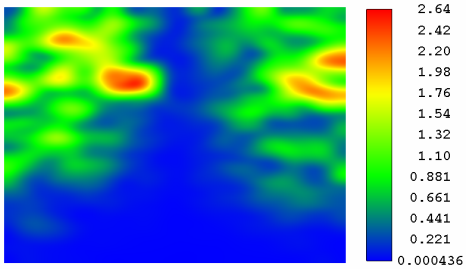
Mode-2 with 140% pink noise



Mode-2 with 175% pink noise

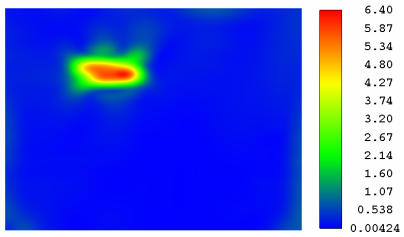


Mode-2 with 210% pink noise

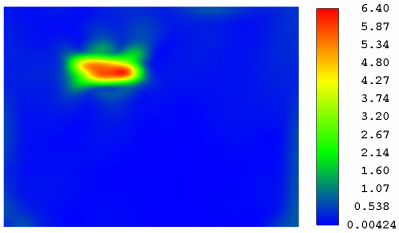


Mode-2 with 245% pink noise

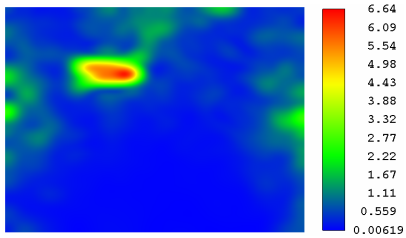
Figure 5.28.—Mode-2 with pink noise



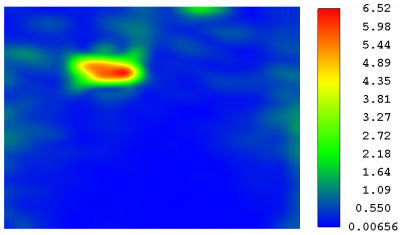
Mode-3 with 0% white noise



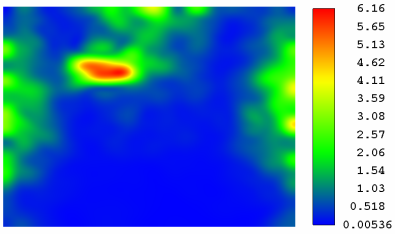
Mode-3 with 0% pink noise



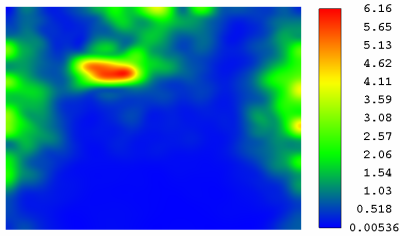
Mode-3 with 10% white noise



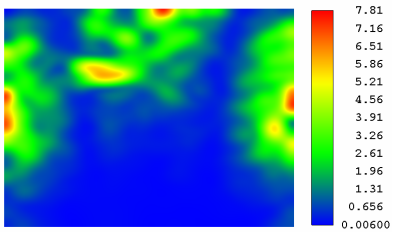
Mode-3 with 10% pink noise



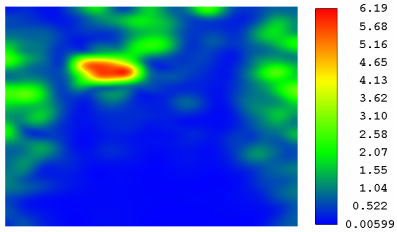
Mode-3 with 17.5% white noise



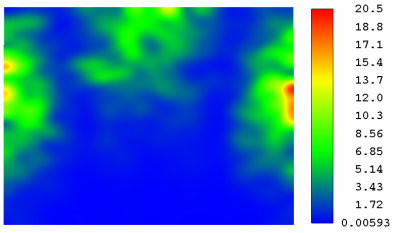
Mode-3 with 17.5% pink noise



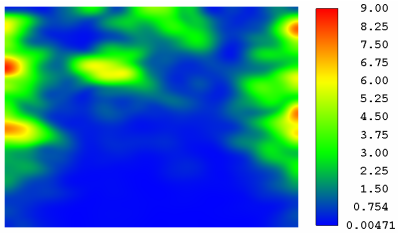
Mode-3 with 35% white noise



Mode-3 with 35% pink noise

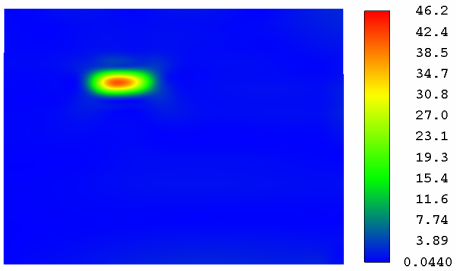


Mode-3 with 70% white noise

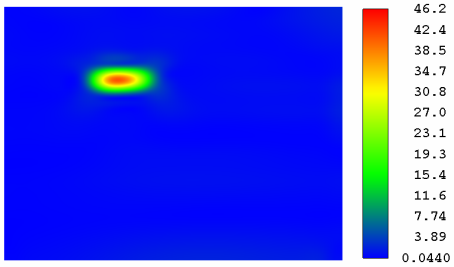


Mode-3 with 70% pink noise

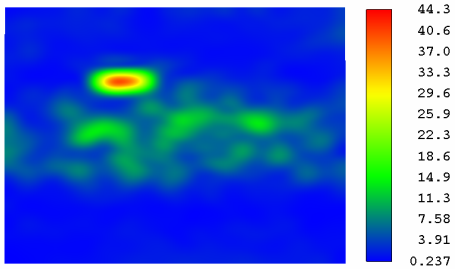
Figure 5.29.—Mode-3 with white noise and pink noise.



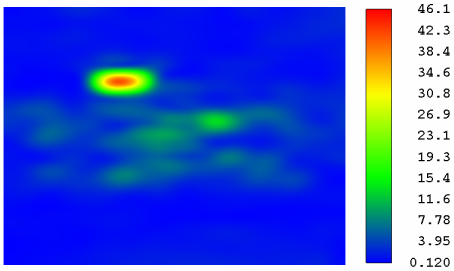
Mode-4 with 0% white noise



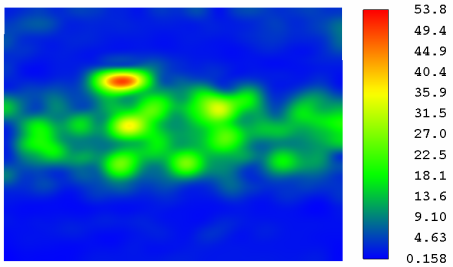
Mode-4 with 0% pink noise



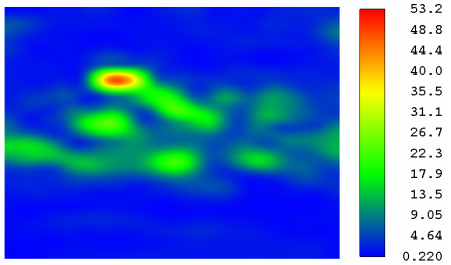
Mode-4 with 35% white noise



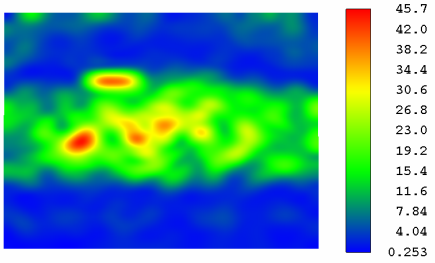
Mode-4 with 35% pink noise



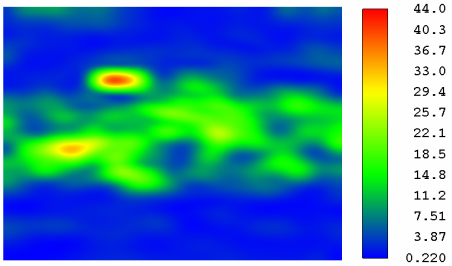
Mode-4 with 70% white noise



Mode-4 with 70% pink noise

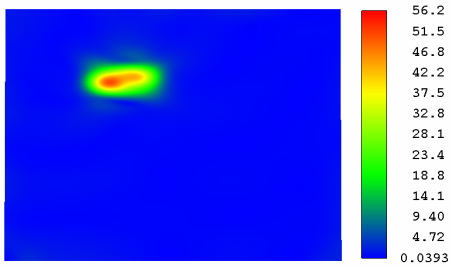


Mode-4 with 105% white noise

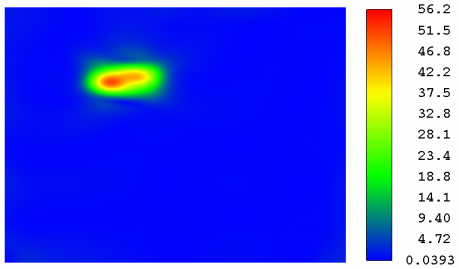


Mode-4 with 105% pink noise

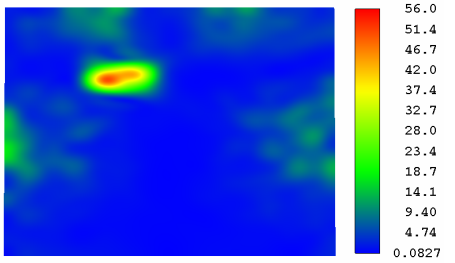
Figure 5.30.—Mode-4 with white noise and pink noise.



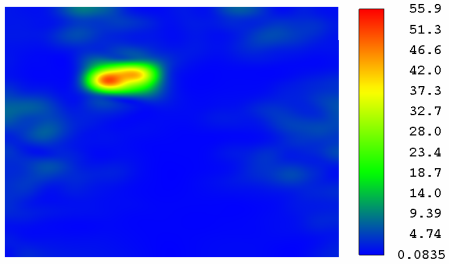
Mode-5 with 0% white noise



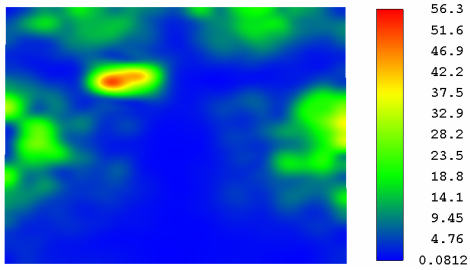
Mode-5 with 0% pink noise



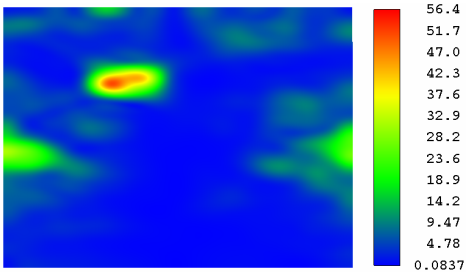
Mode-5 with 35% white noise



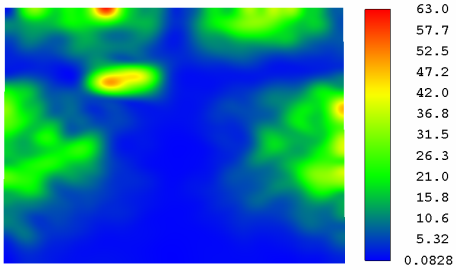
Mode-5 with 35% pink noise



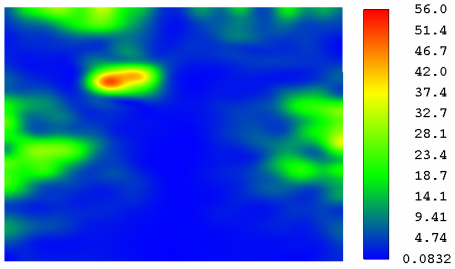
Mode-5 with 70% white noise



Mode-5 with 70% pink noise



Mode-5 with 105% white noise



Mode-5 with 105% pink noise

Figure 5.31.—Mode-5 with white noise and pink noise.

TABLE 5.9.—ROBUSTNESS OF GLOBAL DAMAGE DETECTION WITH WHITE NOISE AND PINK NOISE

Out–of–plane	Mode-1	Mode-2	Mode-3	Mode-4	Mode-5
White noise level	Up to 175%	Up to 105%	Up to 17.5%	Up to 70%	Up to 70%
Pink noise level	Up to 175%	Up to 175%	Up to 35%	Up to 105%	Up to 105%

From table 5.9, we can say that, if one can provide several modes, this detection scheme can able to withstand up to 70 percent of noise.

5.2.7 Signal – Feature Reconstruction.—To study signal feature reconstruction, white noise is added to the measurements in two ways.

1.) Noise is added to the displacements in order to see its effect on the damage parameter and the subsequent reconstruction of the damage parameter feature using Wavelet Toolbox in Matlab.

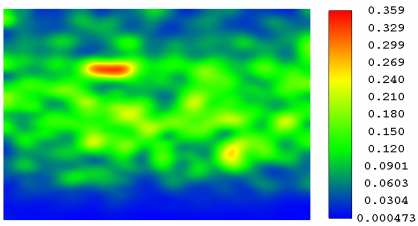
2.) Noise is added to the damage parameter values and the reconstruction damage parameter feature using Wavelet Toolbox in Matlab. The following equation is used to add the noise in the damage parameter values.

$$P_n = P + \max(P)*\%\text{ of } N$$

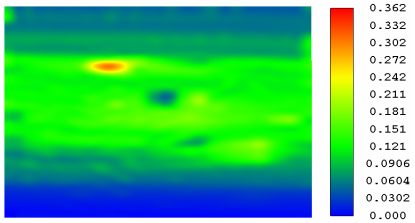
where,

P_n = Noised damaged Parameter
P = Ideal Damage Paramter
N = Noise = rand(-1, 1)

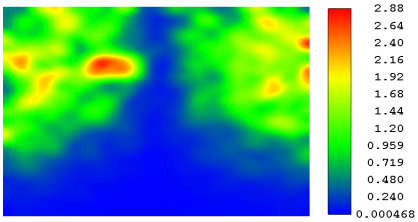
Ideal (noise free) case is considered from the section 5.2.1. Results of case-1, which is the addition of the noise to the displacements, simulations are presented in figure 5.32 and results of case-2, which is the addition of the noise to the damage parameter values, simulations are presented in figure 5.33.



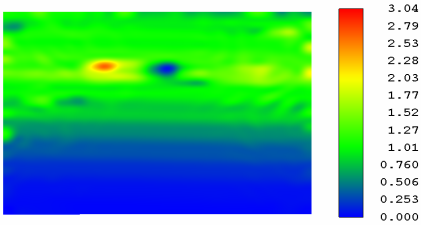
Mode-1 with 210% white noise



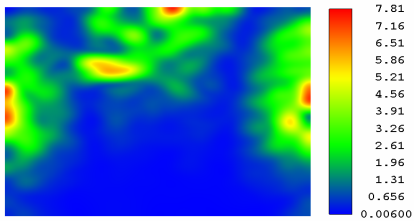
Mode-1 denoised



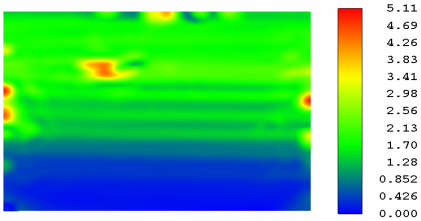
Mode-2 with 175% white noise



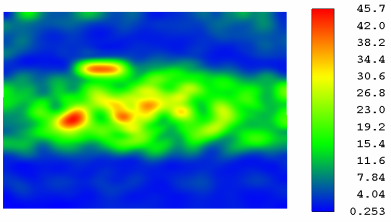
Mode-2 denoised



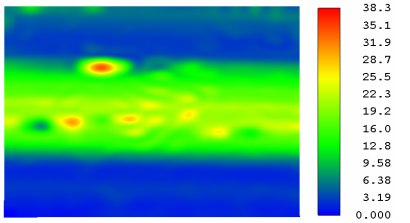
Mode-3 with 35% white noise



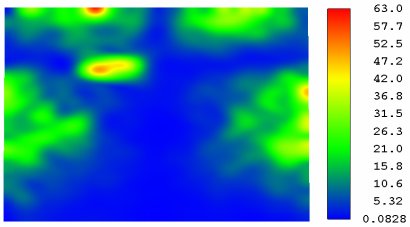
Mode-3 denoised



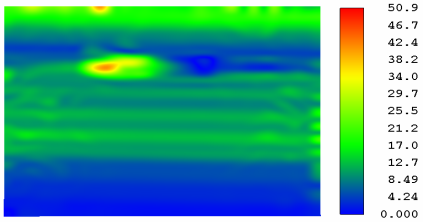
Mode-4 with 105% white noise



Mode-4 denoised



Mode-5 with 105% white noise



Mode-5 denoised

Figure 5.32.—Reconstruction of damage parameter with white noise added to the displacements.

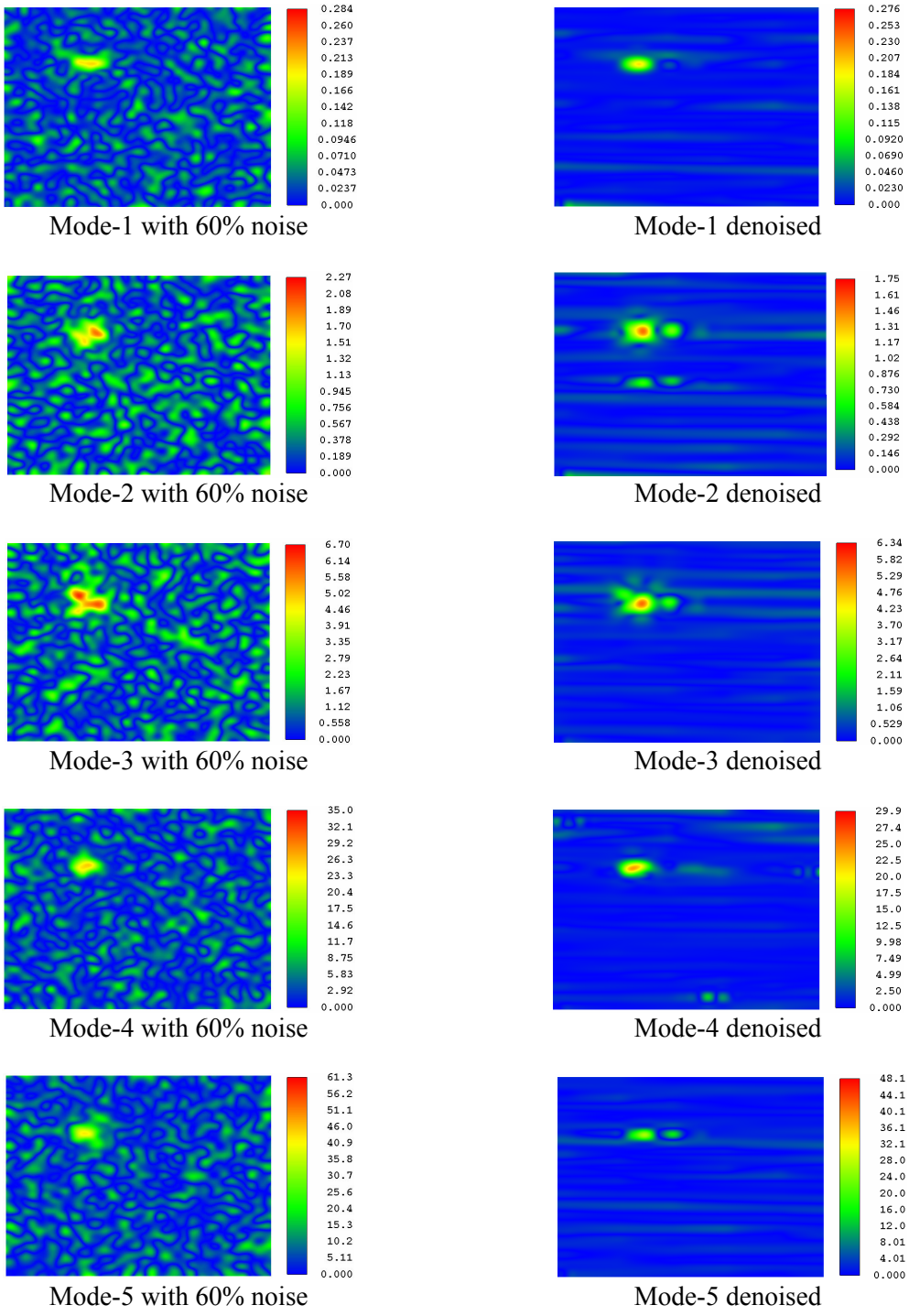


Figure 5.33.—Reconstruction of damage parameter with white noise added to the damage parameter values.

An additional case is considered which is unlike to the previous cases. Here, instead of taking a single damage, two damages are considered on a plate shown in figure 5.34, one is oriented horizontally and other is oriented at 45°. The plate is simply supported and it has the same dimensions and material properties as in the section 5.2.1. Modulus of elasticity is reduced by 66 percent for both of the damages. The five fundamental mode shapes are extracted from the dynamic analysis and the corresponding undamaged frequencies are listed in table 5.10. The white noise is added to the damage parameter to study the feature reconstruction capability in global damage detection. Mode shapes and damage parameter contours are presented in figure 5.35. Figure 5.36 shows the noised and the reconstructed damage parameter contours.

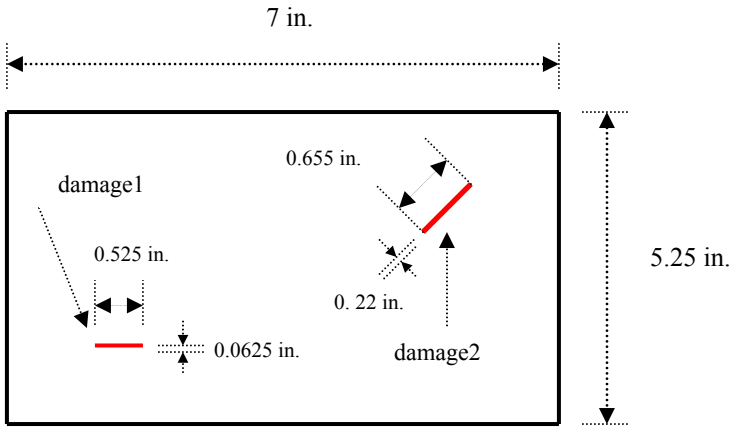


Figure 5.34—Simply supported plate with two damages.

Location of damage1 from left bottom corner: 1.05 in. in *x*-dir and 1.05 in. in *y*-dir
Location of damage2 from right top corner: 1.225 in. in *x*-dir and 1.31 in. in *y*-dir

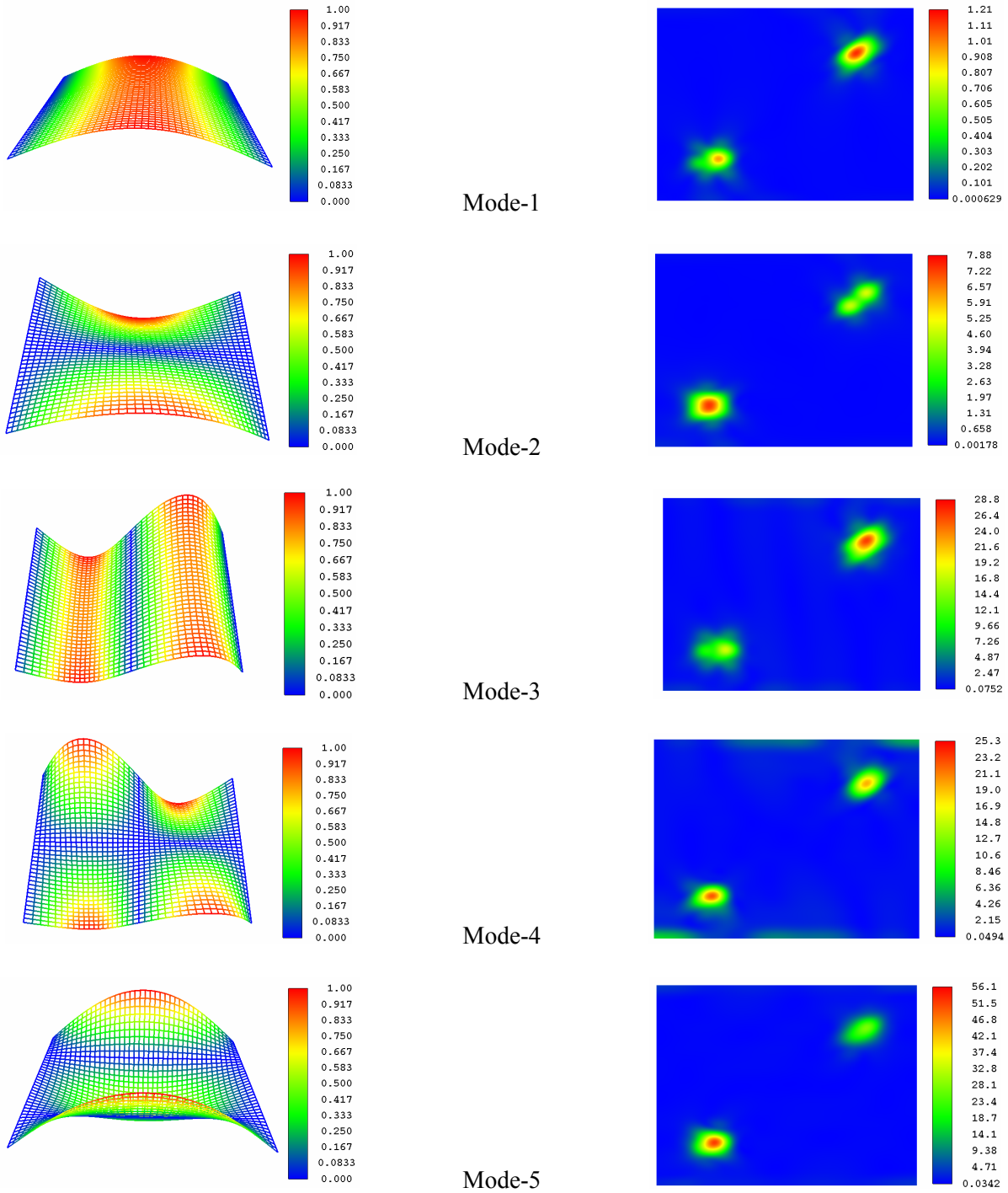


Figure 5.35.—Simply supported plate with two damages oriented horizontally and inclined at 45° (a) Mode shapes (b) Damage parameter (noise free).

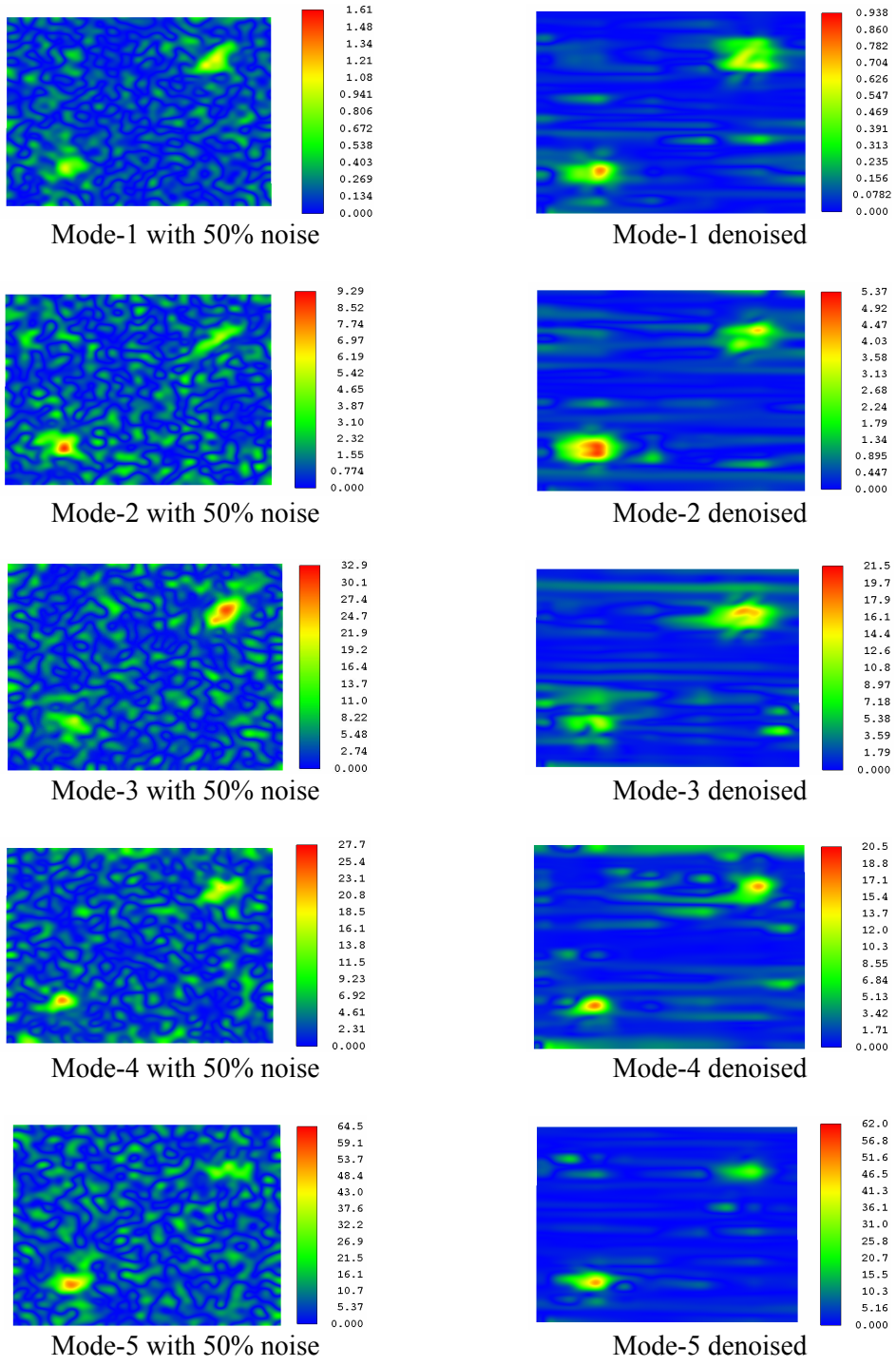


Figure 5.36.—Reconstruction of damage parameter with white noise added to damage parameter values with two damages oriented horizontally and inclined at 45°.

TABLE 5.10.—UNDAMAGED FREQUENCIES (CYC/SEC)
OF SIMPLY SUPPORTED PLATE

Mode-1	231.0301748
Mode-2	473.1115781
Mode-3	934.7355844
Mode-4	1245.773229
Mode-5	1316.060446

5.2.8 False Alarm Test.—A false alarm case study is performed to investigate the proposed detection scheme for this research. A simply supported plate of 7 in. by 5.25 in. by 0.125 in./is considered. Two areas are considered for damage, one is oriented horizontally and other is oriented at 45°. Figure 5.34 shows the size and locations of the damages. For the false alarm test, at first the structure is weakened by 66 percent and in the second test the structure is strengthened by 66 percent. Damage vector fields are extracted to distinguish the weakening and strengthening of the structure at a specific location. The damage vector points in the direction of decreased total energy. Figure 5.52 shows the weakening of the structure, where the damage vectors are directed outward from the location. Figure 5.53 shows the strengthening of the structure, where the damage vectors are directed inward at the location.

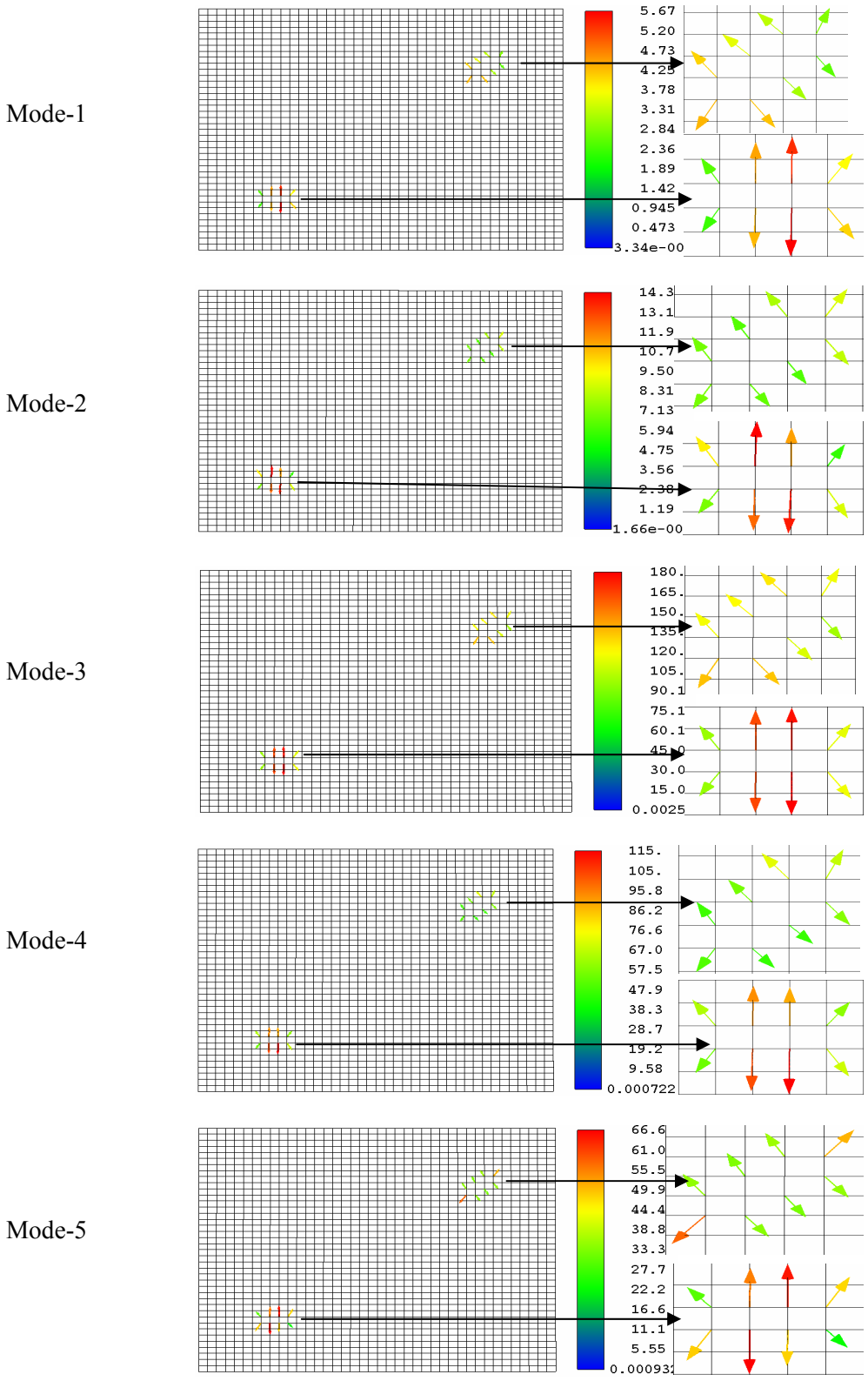


Figure 5.37.—Damage nodal force vectors for weakening of the structure.

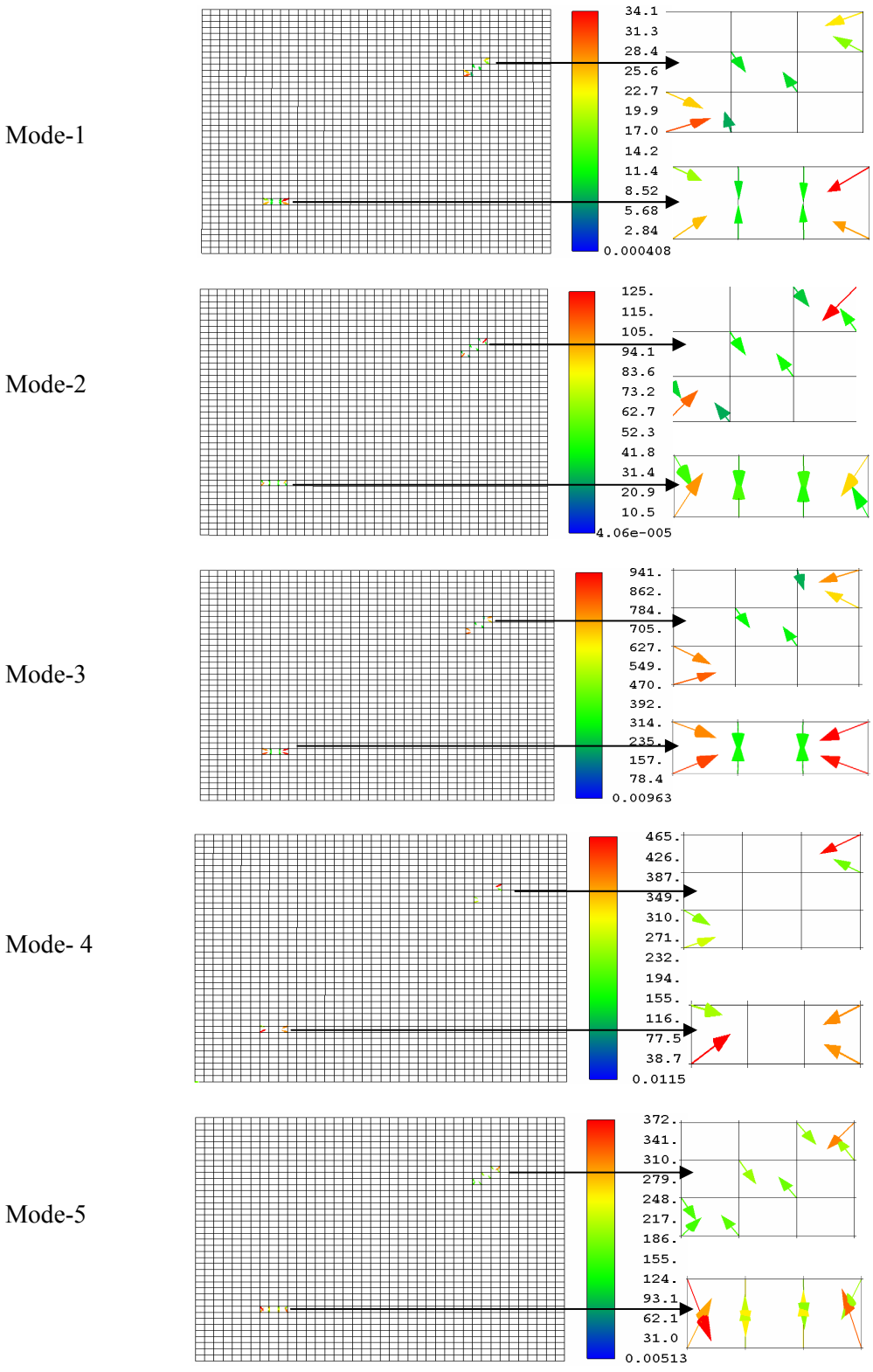
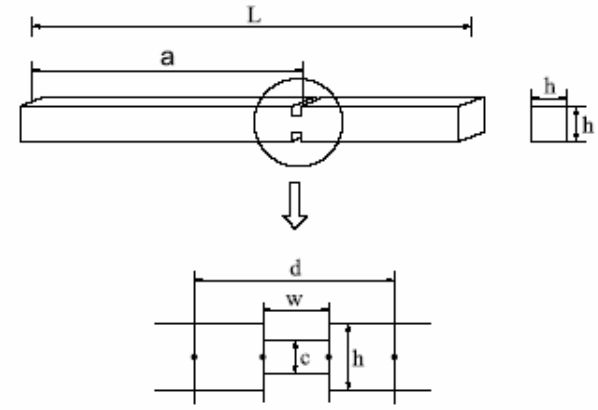


Figure 5.38.—Damage nodal force vectors for strengthening of the structure.

5.2.9 A Comparative Case Study.—J.-C. Hong, et al. (ref. 48) described how wavelet transforms can be adopted for damage detection using the vibration modes of beam. They suggested Continuous Wavelet Transforms (CWT) with the Mexican-hat wavelet found in the Matlab wavelet toolbox. The Mexican-hat wavelet function is the second derivative of the Gaussian probability density function. They also used ANSYS to simulate a simply supported beam model with 2400 two-node beam elements (element size = 0.5 mm). Location of the damage is at a = 800 mm. Material properties are Young's Modulus = 70 GPa, Density = 2700 kg/m^3.

The model for the analysis is shown in figure 5.37. Modal analysis is performed in ANSYS and the first bending mode is passed through the wavelet toolbox. The wavelet analysis has detected the damage and has located it exactly at the correct position. Figure 5.39 shows the contour plot of the wavelet transform in which we can observe the location of the damage.

A similar case study is performed with the simulations described in the literature using ABAQUS. figure 5.41, which is a contour of wavelet analysis, shows the location of the damage. Here, the wavelet analysis utilized an extensive (2400 points) sensor network. As we can clearly observe, there is a measurement (sensor) right on top of the damage. In reality, there is a small chance that there will be a measurement right on top of the damage since location of the damage is unknown and arrangement of extensive sensor networks also very difficult. Therefore, coarser network is considered. Instead of passing 2400 data points, in first case, only 100 points are passed (a measurement after every 24 points from 2400 data points) and in second case 160 data pointes are passed (a measurement after every 15 points from 2400 data points). In both cases, the direct wavelet approach is not able to detect the damage. The results are presented form figures 5.42 to 5.45.



(L=1200 mm, h=20mm, w=0.5 mm)

Figure 5.39.—Model for the simulation.

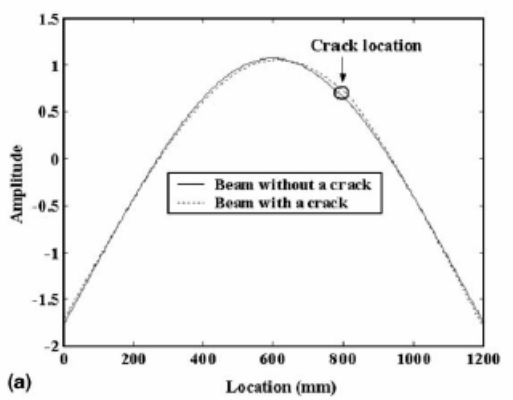


Figure 5.40.—First bending mode from the literature.

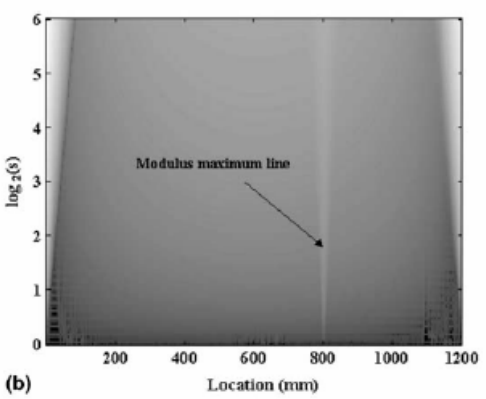


Figure 5.41.—Contour plot of wavelet transform from the literature.

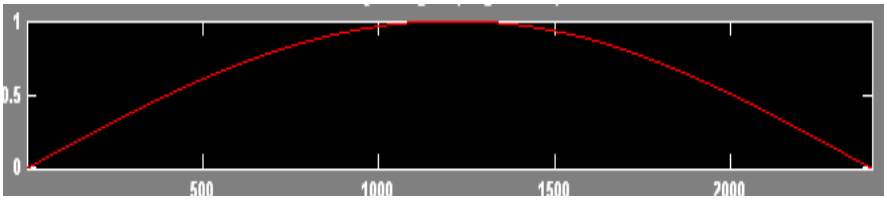


Figure 5.42.—First bending mode.

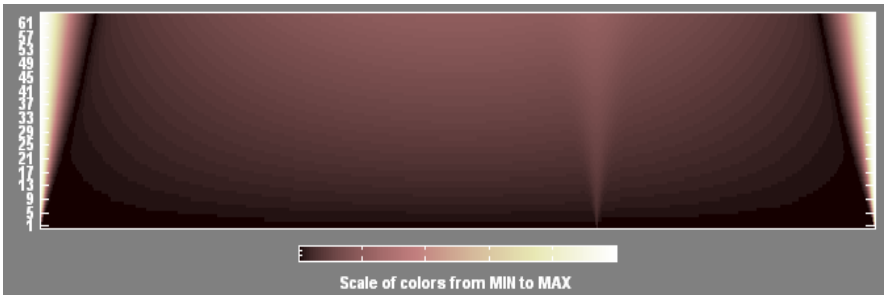


Figure 5.43.—Contour plot of wavelet transform.

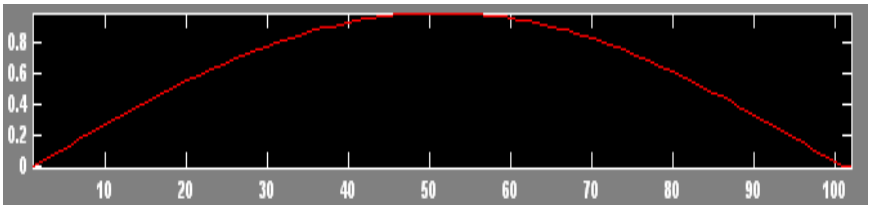


Figure 5.44.—First bending mode with 100 data points.

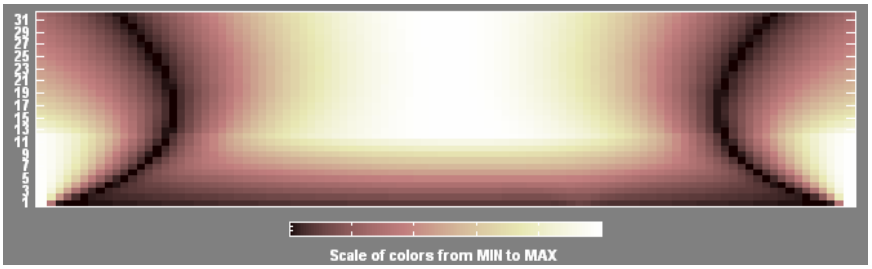


Figure 5.45.—Contour plot of wavelet transform with 100 data points.

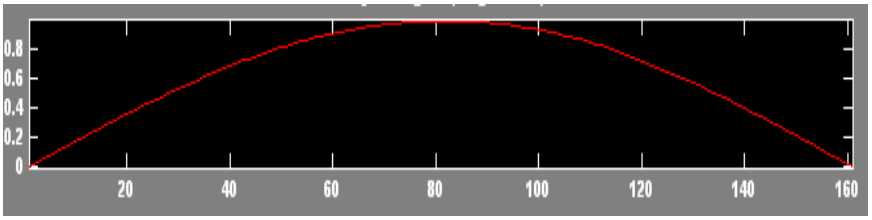


Figure 5.46.—First bending mode with 160 data points.

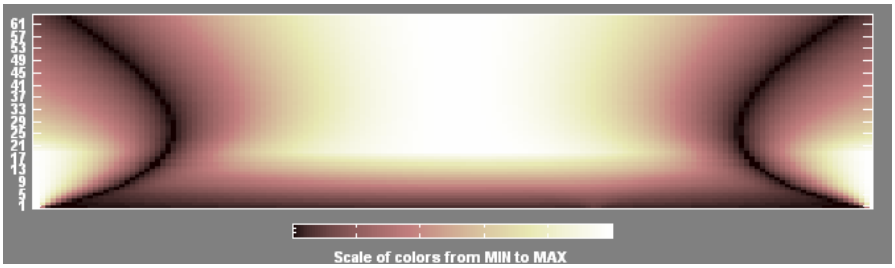


Figure 5.47.—Contour plot of wavelet transform with 160 data points.

An equivalent simulation is performed using the damage detection scheme proposed in this report for a simply supported beam of same dimensions. Damage is simulated over a 0.1 percent area of the beam by degradation of the modulus of elasticity by 14 percent at the location of the damage. Instead of passing the mode shape to the wavelet analysis, the damage parameter is passed which is derived from the proposed scheme. Figure 5.47 shows the contour of wavelet analysis. Length *l* is the approximate size of the damage shown in the contour with the displacements measured at 600 points. In addition, the damage parameter is calculated for coarser networks also. The displacements are considered at 20 and 10 points to simulate the coarser networks. Care was taken such that there is no measurement at the location of the damage. In both coarser networks, the damage was successfully detected. If we observe the wavelet analysis contour plots in figures 5.49 and 5.51, the size of the damage is wider than in the figure 5.47, since it depends on the element size.

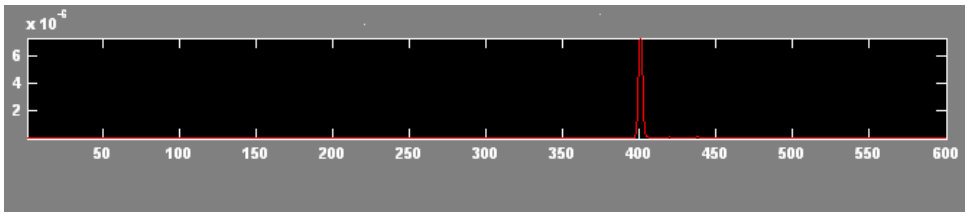


Figure 5.48 Damage Parameter in first bending mode with 600 data points.

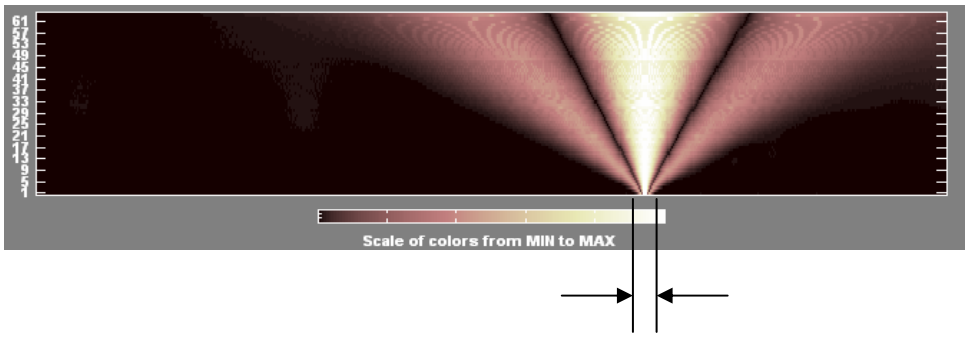


Figure 5.49.—Contour plot of wavelet transform with 600 data points.

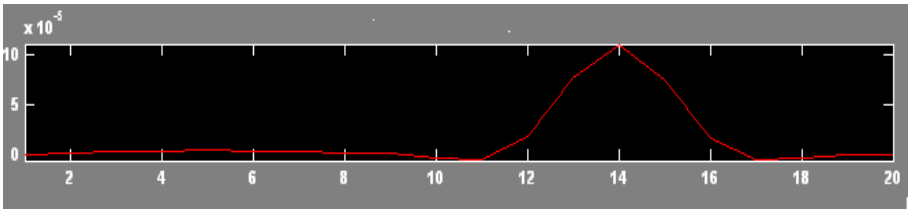


Figure 5.50.—Damage Parameter in first bending mode with 20 data points.

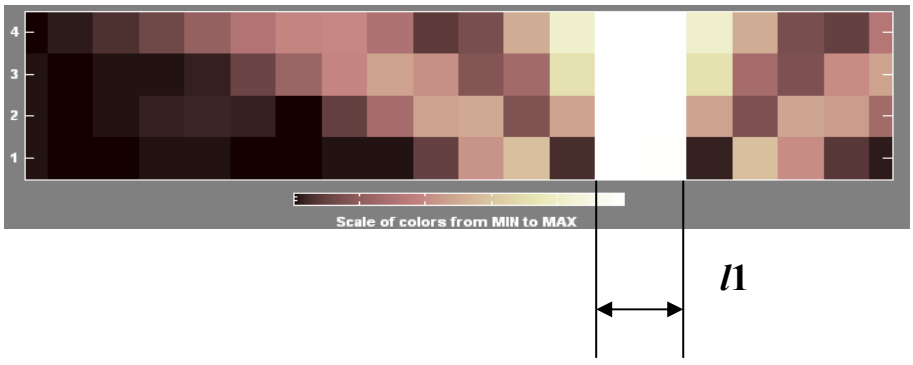


Figure 5.51.—Contour plot of wavelet transform with 20 data points.

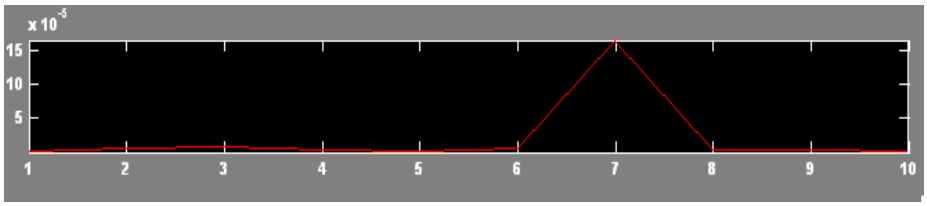


Figure 5.52.—Damage Parameter in first bending mode with 10 data points.

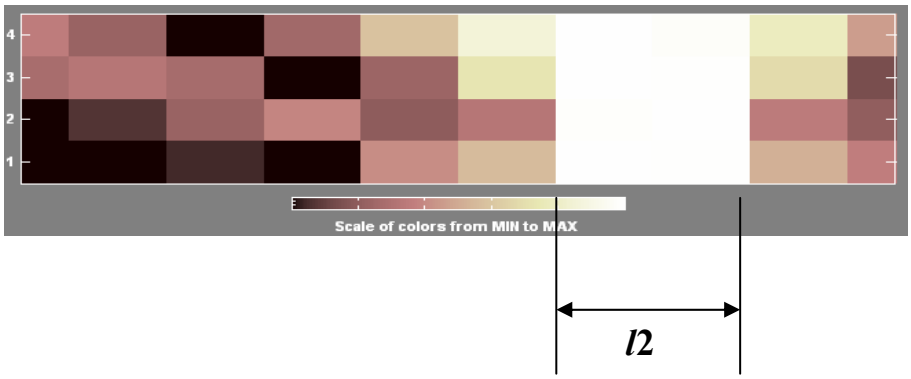


Figure 5.53.—Contour plot of wavelet transform with 10 data points.

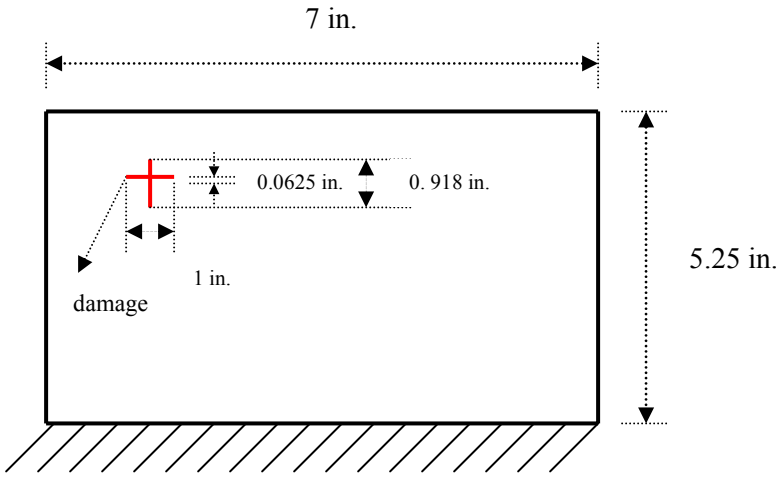
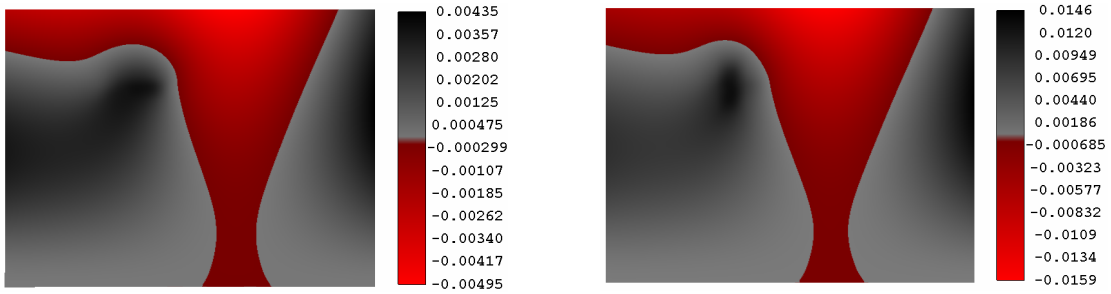


Figure 5.54.—Cantilever plate with 12 damaged elements.

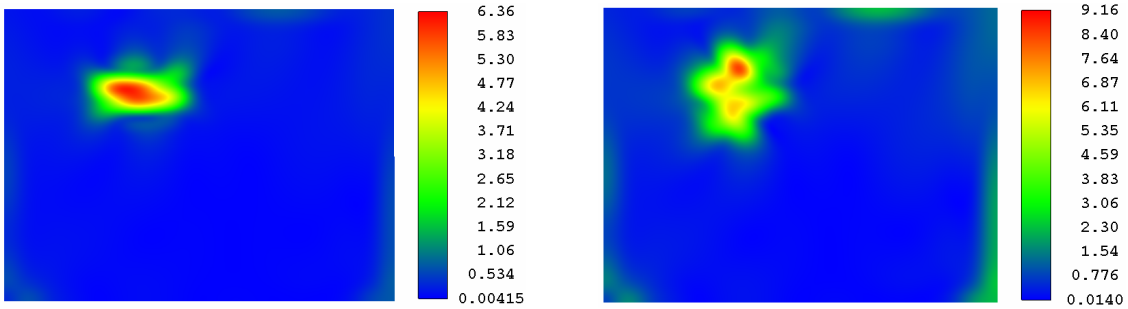
5.2.10 A Case Study on the Severity of the Damage.—A robust damage detection technique should be able to judge the severity of the damage. To study the severity of the damage two different damage sizes are considered. One size of the damage has considered from the section 5.2.1. To simulate more severe damage, double the size of the damage in the section 5.2.1 is considered. Including the six damage elements in section 5.2.1, six more elements are damaged. Figure 5.52 shows the size and shape of the damage. The mode 3, which is the mixture of bending and twisting in cantilever plate, is considered to investigate the severity.

Figure 5.53 shows the delta mode shape, which the difference between the damaged and the undamaged mode shapes, wherein the difference is greater in the double the size of the damage than the actual size. Figure 5.54 shows the damage parameter contour, wherein the magnitude of the double the size of the damage is greater then the actual size of the damage. The two figures show the severity of the damage. Therefore, we can judge the severity of the damage by inspecting the delta mode shape and the damage parameter contours.



(a) (b)

Figure 5.55.—Delta mode shapes (a) Damage with actual size (b) Damage with double the size.



(a) (b)

Figure 5.56.—Damage parameter (a) Damage with actual size (b) Damage with double the size.

Chapter VI
Summary, Conclusions and Future Work

6.1 Summary

The presented study was conducted in conjunction with an ongoing research program dealing with the development of an integrated analytical/experimental nondestructive evaluation (NDE) methodology for global damage detection. The important ingredient is the use of a damage energy parameter. It provides the required sensitivity measure for detection of the damage on the basis of measured structural response.

From the viewpoint of practical utilization in structures, extensive testing for a valid assessment of any NDE technique becomes necessary. Especially in view of the many complicating factors and the variety of damage scenarios that are likely to be encountered, assessment of the scheme becomes a necessary requirement. In the real time environment, damage can manifest itself as: weakened material, a line crack, or corrosion. An investigation is performed keeping the above types of damage in mind. A case study is conducted on the density of the arrangement of the sensor network. The extensive/fine sensor network has unlimited capability of measuring response at all regions in the structure. However, from a practical viewpoint, it is very difficult to arrange the extensive network due to various factors. Therefore, effectiveness of the scheme is tested with coarse sensor networks. In addition, by observing the difficulty in obtaining noise free measurements in the real time environment, a case study is performed to assess the capability of the scheme to detect damage at various noise levels. If the technique is excessively sensitive to the noise levels then the effect of the "true" damage will be completely "overshadowed" by the noise. That means, extracting the damage feature will be a difficult task. Therefore, feature reconstruction plays an important role in global damage detection. In this report, the feature extraction capability of the detection scheme is investigated extensively. With an eye towards reducing the noise intensity, the development of different types of filters has been undertaken. Here, wavelet theory is utilized to reconstruct the damage feature from the noise to enhance the feature extraction. An important factor in global damage detection is to over come the false alarm, i.e., the detection scheme should able to detect whether there is a weakening or a strengthening of the structure. As a result, a false alarm test is conducted on the detection scheme.

6.2 Conclusions

Based on the presented results, the following conclusions can be made:

1. The detection scheme is able to detect and locate any type of damage (i.e., the different failure modes) present in the structure. As expected, the so-called delta mode shapes are completely different than the actual mode shapes.
2. The required features are extracted efficiently by the Eview graphical program, which allows the user to view the large and sometimes complex files generated by the proposed detection scheme.
3. The detection scheme is efficient enough to extract the feature of the damage even with coarse sensor networks.
4. When actual preliminary experimental files (using Electronic Speckle Pattern Interferometry (ESPI)) were analyzed, the delta mode shapes behaved similar to the actual mode shapes. These experimental measurements were determined not to be directly usable for the purpose of detecting small amounts of damages. As they presently exist, the ESPI techniques focus on the determination of correct overall underlying mode shapes and frequencies. Recall that for the small amounts of damages targeted in the present study, essentially identical overall modes of vibration were obtained before and after damage (see figs. 5.3(a) and (b)).This indeed is one of the main motivations for conducting the noise level quantification studies reported here.

5. When the noise is added to the displacements, i.e., before processing the data, the damage detection scheme has been shown to be able to withstand up to 70 percent of noise (see table 5.9) when several modes are provided for detection.

6. When noise is added to the damage parameter, i.e., after processing the data, the damage detection scheme has shown to be able to withstand up to 50 percent of noise (see figs. 5.33 and 5.36).

7. A one dimensional wavelet analysis is used for the feature reconstruction of the damage parameter signal from the noise. Wavelet transforms greatly reduced the intensity of the noise by retaining the significant damage values.

8. The detection scheme is able to distinguish the strengthening and the weakening of the structure, i.e., it overcame the false alarm test.

9. We can conclude that, if ideal measurements are provided, even though there may be few measurements, i.e., on coarser sensor network, the proposed scheme can detect and locate the damage. Also, the scheme can withstand up to a level of 70 percent of noise when raw displacements are directly utilized.

10. However, utilizing further enhancement through wavelet tools, i.e., denoising/ feature reconstruction, one is able to use the detection scheme with 105 percent of noise in the displacements when several modes are provided.

11. Enhancement of the damage feature extraction capability by reducing the intensity of the noise is achieved using the wavelets (see figs. 5.32, 5.33, and 5.36).

References

1. Chen, J.-C., and garba, J.A., 1988, "On-Orbit Damage Assessment for Large Space Structures," *AIAA Journal*, vol. 26, no. 9, 1119–1126.
2. McGowan, P.E., Smith, S.W., and Javeed, M., 1990, "Experiments for Locating Damage Members in a Truss Structures," Proc. 2nd USAF/NASA Workshop on system Identification and Health Monitoring of Precision Space Structures, 571–615.
3. Smith, S.W., 1992, "Iterative Use of Direct Matrix Updates: Connectivity and Convergence," in proc. Of 33rd AIAA Structures, Structural Dynamics and Material Conference, 1797–1806.
4. Kim, H.M., and Bartkowicz, T.J., 1993, "Damage Detection and Health Monitoring of Large Space Structures," *Sound and Vibration*, vol. 27, no. 6, 12–17.
5. Lindner, D.K., Twitty, G.B., and Osterman, S., 1993, "Damage Detection for Composite Materials using Dynamic Response Data," *ASME Adaptive Structures and Materials Systems,* AD 35, 441–448.
6. Liu, P.-L., 1995, "Identification and Damage Detection o Trusses using Modal Data," *Journal of Structural Engineering,* vol. 121, no.4, 599–608.
7. Zimmerman, D.C., and Kaouk, M., 1994, "Structural Damage Detection Using a Minimum Rank Update Theory," *Journal of Vibration and Acoustics*, vol. 116, 222–230.
8. Kaouk, M., and Zimmerman, D.C., 1994, "Structural Damage Assessment Using a Generalized Minimum Rank Perturbation Theory," *AIAA Journal,* vol. 32, no. 4, 836–842.
9. Kaouk, M., and Zimmerman, D.C., 1994, "Assessment of Damage Affecting All Structural Properties," in Proc. of the 9th VPI &SU Symposium on Dynamics and Control of Large Structures, 445–455.
10. Kaouk, M., and Zimmerman, D.C., 1994, "Structural Damage Detection Using Measured Modal Data and No Original Analytical Model," in proc. of the 12th International modal Analysis Conf., 731–737.
11. Zimmerman, D.C., Kaouk, M., and Simmermacher, T., 1995, "Structural Damage Detection Using Frequency Response Functions," in Proc. of the 13th International Modal Analysis Conf., 179–184.
12. Zimmerman, D.C., Kaouk, M., and Simmermacher, T., 1995, "On the Role of Engineering Insight and Judgement Structural Damage Detection," in Proc. of the 13th International Modal Analysis Conf., 414–420.

13. Ricles, J.M., 1991, "Nondestructive Structural Damage Detection in Flexible Space Structures Using Vibration Characterization," NASA report CR–185670.
14. Sanayei, M., and Onipede. O., 1991, "Damage Assessment of Structures Using Static Test Data," *AIAA Journal*, vol. 29, no. 7, 1174–1179.
15. Sanayei, M., Onipede, O., and Babu, S.R., 1992, "Selection of Noisy Measurement Locations for Error Reduction in Static Parameter Identification'" *AIAA Journal,* vol. 30, no. 9, 2299–2309.
16. Hemez, F. M., and Farhat, C., 1995, "Structural Damage Detection via a Finite Element Model Updating Methodology," *Modal Analysis—The International of Analytical and Experimental Modal Analysis,* vol. 10, no. 3, 152–166.
17. Zimmerman, D.C., and Kaouk, M., 1992 "Eigenstructure Assignment Approach for Structural Damage Detection," *AIAA Journal*, vol. 30, no. 7, 1848–1855.
18. Linder, D.K., and Goff, R., 1993, "Damage Detection, Location and Estimation for Space Trusses," SPIE *Smart Structures and Intelligent Systems,* 1028–1039.
19. Schulz, M.J., Pai, P.F., and Abdelnaser, A.S., 1996, "Frequency Response Function Assignment Technique for Structural Damage Identification," in Proc. Of the 14th International Modal Analysis Conference, 105–111.
20. Naruh, H., and Ratan, S., 1993, "Damage Detection in Flexible Structures," *Journal of Sound and Vibration*, vol. 166, no. 1, 21–30.
21. Kim, H.M., and Bartkowicz, T.J., 1994, "A Two-Step Structural Damage Detection Approach with Limited Instrumentation," *Journal of Vibration and Acoustic—Transactions of the ASME*, vol. 119, no. 2, 258–264.
22. Kim, H.M., Bartkowicz, T.J., Smith, S.W., and Zimmerman, D.C., 1995, "Structural Health Monitoring of Large Structures," in Proc. of 49th Meeting of the Society for Machinery Failure Prevention Technology, 403–412.
23. Li, C., and Smith, S.W., 1995, "Hybrid Approach for Damage Detection in Flexible Structures," *Journal of Guidance, Control, and Dynamics,* vol. 18, no. 3, 419–425.
24. Hung-Liang, R.C., and Kiriakidis, A.C., 2000, "Stiffness Evaluation and Damage Detection of Ceramic Candle Filters," *Journal of Engineering Mechanics,* vol. 126, no. 3, 308–319.
25. Maeck, J., Waheb, M.A., Peeters, B., DeRoeck, G., DeVisscher, J., DeWilde, W.P., Ndambi, J.-M., and Vantomme, J., 2000, "Damage Identification in Reinforced Concrete Structures by Dynamic Stiffness Determination," *Engineering Structures*, vol. 22, 1339–1349.
26. Yuen, M.M.F., 1985, "A Numerical Study of The Eigenparameters of A Damaged Cantilever," *Journal of Sound and Vibration*, vol. 103, no. 3, 301–310.
27. Ren, W.-X., and DeRoeck, G., 2002, "Structural Damage Identification Using Modal Data – I: Simulation Verification," *Journal of Structural Engineering*, vol. 128, no. 1, 87–95.
28. Ren, W.-X., and DeRoeck, G., 2002, "Structural Damage Identification Using Modal Data – II: Test Verification," *Journal of Structural Engineering,*vol. 128, no. 1, 96–104.
29. Stubbs, N., and Osequeda, R., 1990, "Global Damage Detection in Solids: Experimental Verification," *International Journal of Analytical and Experimental Modal Analysis*, vol. 5, no. 2, 81–97.
30. Gawronski, W., and Sawicki, J.T., 2000, "Structural Damage Detection Using Modal Norms," *Journal of Sound and Vibration,* vol. 229, no. 1, 194–198.
31. Rizos, P. F., Asparagathos, N., Dimarogonas, A. D., 1990, "Identification of Crack Location and Magnitude in a Cantilever Beam for the Vibration Modes," *Journal of Sound and Vibration*, vol. 138, no. 3, 381–388.
32. Vestroni, F., and Capeecchi, D., 2000, "Damage Detection in Beam Structures Based on Frequency Measurements," *Journal of Engineering Mechanics*, vol. 126, no. 7, 761–768.
33. Stubbs, N., Kim, J.-T., Topole, K., 1992, "An Efficient and Robust Algorithm for Damage Localization in Offshore Platforms," in *Proc. ASCE Tenth Structures Congress*, 543–546.
34. Topole, K.G., and Stubbs, N., 1995, "Nondestructive Damage Evaluation of a Structure from Limited Modal Parameters," *Earthquake Engineering and Structural Dynamics*, vol. 24, no. 11, 1427–1436.

35. Topole, K.G., and Stubbs, N., 1995, "Nondestructive Damage Evaluation in Complex Structures from Minimum of Modal Parameters," *Modal Analysis – The International Journal of Analytical and Experimental Modal Analysis*, vol. 10, no. 2, 95–103.
36. Stubbs, N., and Kim, J.T., 1996, "Damage Localization in Structures without Baseline Modal Parameters," *AIAA Journal,* vol. 34, no. 8, 1644–1649.
37. Pandey, A.K., Biswas, M., and Samman, M.M., 1991, "Damage detection from Changes in Curvature Mode Shapes," *Journal of Sound and Vibration*, vol. 145, no. 2, 321–332.
38. Chance, J., Tomilinson, G.R., and Worden, K., 1994, "A Sinplified Approach to The Numerical and Experimental Modeling of The Dynamics of a Cracked Beam," in Proc. of the 12th International Modal Analysis Conference, 778–785.
39. DisPasquale, Edmondo, Jiaun-Wen Ju, Attila Askar, and Ahmed S. Cakmak, 1990, "Relation Between Global Damage Indicies and Local Stiffness Degradation," *Journal of Structural engineering*, vol. 116, no. 5, 1440–1456.
40. Stubbs, N., and Osegueda, R., 1990, "Global Non-Destructive Damage Evaluation in Solids," *The International Journal of Analytical and Experimental Modal Analysis*, vol. 5, no. 2, 67–79.
41. Stubbs, N., and Osegueda, R., 1990, "Global Damage Detection in Solids-experimental Verification," *The Journal of Analytical and Experimental Modal Analysis*, vol. 5, no. 2, 81–97.
42. West, W.M., 1984, "Illustration of The Use of Modal Assurance Criterion to Detect Structural Changes in an Orbiter Test Specimen," in Proc. Air Force Conference on Aircraft Structural Integrity, 1–6.
43. Mayer, R.L., 1992, "Error Localization using Mode Shapes—An Application to A Two Link Robot Arm," in Proc. 10th International Modal Analysis Conference, 886–891.
44. Ratcliffe, C.P., 1997, "Damage Detection using a Modified Laplacian Operator on Mode Shape Data," *Journal of Sound and Vibration*, vol. 204, no. 3, 505–517.
45. Cobb, R.G., and Liebst, B.S., 1997, "Sensor Placement and Structural Damage Identification from Minimal Sensor Information," *AIAA Journal*, vol. 35, no. 2, 369–374.
46. Stubbs, N., and Osegueda, R. 1990, "Global Non-Destructive Damage Evaluation in Solids," *International Journal of Analytical and Experimental Modal Analysis,* vol. 5, no. 2, 67–79.
47. Hou, Z., Noor, M., and St. Amand, R., 2000, "Wavelet – Based Approach for Structural Damage Detection," *Journal of Engineering Mechanics,* vol. 227, no. 7, 677–683.
48. Hong, J.-C., Kim, Y.Y., Lee, H.C., and Lee, Y.W., 2002, "Damage Detection Using The Lipschitz Exponent Estimated by The Wavelet Transform: Application to Vibration Modes of a Beam," *International Journal of Solids and Structures*, vol. 39, 1803–1816.
49. Yan, Y.J., Hao, H.N., and Yam, L.H., 2004, "Vibration-based Construction and Extraction of Structural Damage Feature Index," *International Journal of Solids and Structures*, vol. 41, 6661–6676.
50. Ikumasa Yoshida, and Tadanobu Sato, 2002, "Identification of Damping Ration using Monte Carlo Filter Based on Exclusive Non-Gaussian Process Noise," in Proc. 15th ASCE Engineering Mechanics Conference, 1–8.
51. Taeho Charles Jo, 1997, "MLP Training with Noise Optimized by Genetic Algorithm," in the workshop of EUROGEN 97.
52. Xu, Y. G., and Liu, G. R., 2002, "Detection of Flaws in Composites from Scattered Elastic-Wave Field suing an Improved μ GA and a Local Optimizer," *Journal of Computer Methods in Applied Mechanics and Engineering,* vol. 191, 3929–3946.
53. Cacciola, P., Impollonia, N., and Muscolino, G., 2003, "Crack Detection and Location in a Damaged Beam Vibrating under White Noise," *Journal of Computes and Structures*, vol. 81, 1773–1782.
54. Prashant, M. Pawar, Ranjan Ganguli, 2003, "Genetic Fuzzy Systems for Damage Detection in Beams and Helicopter Rotor Blades," *Journal of Computer Methods in Applied Mechanics and Engineering,* vol. 192, 2031–2057.
55. John M. Halley, 1996, "Ecology, Evolution, and 1/f-Noise," *TREE*, vol. 11, no. 1, 33–37.

56. Keiji Takagi, 1999, "A Model of 1/f Noise Spectrum Generation in Granular Structures," *Journal of Microelectronics Reliability,* vol. 39, 541–544.
57. Kaulakya, B., and Meskauskas, T., 2000, "Models for Generation 1/f Noise," *Journal of Microelectronics Reliability,* vol. 40, 1781–1785.
58. Rangaraj M. Rangayyan, 2002, "Biomedical Signal Analysis: A Case-Study Approach," New York, IEEE Press.
59. Shunsuke Kishimoto, and Masuo Fukue, 2001, "Generation of Physical Random Digits Using Diode Noise and its Statistical Properties," *Journal of Electronics and Communications in Japan*, vol. 84, no. 11, 29–36.
60. Shan Suthaharan, and Ray, S., 1996, "Generation of Signal-Uncorrelated Noise for Control experiments in Image Restoration," *Pattern Recognition Letters*, vol. 17, 219–230.
61. Thanyawat Pothisiri, and Keith D. Hjelmstad, 2000, "Damage Detection for Spatially Sparse and Noise-Polluted Modal Response," 14th Engineering Mechanics Conference*,* The University of Texas at Austin, Texas.
62. Lakhwinder Kaur, Savita Gupta, and Chauhan, R. C., 2002, "Image Denoising using Wavelet Thresholding," *Indian conf. on computer Vision, Graphics and Image Processing*, Ahmedabad, India.
63. Shi Zhong, and Vladimir Cherkassky, 2000, "Image Denoising using Wavelet Thresholding and Model Selection," in *Proc. IEEE Int. Conf. on Image processing,* Vancouver, BC, Canada.
64. Wang, X. H., Robert S. H. Istepanian, and Yong Hua Song, 2003, "Microarry Image Enhancement by Denoising Using Stationary Wavelet Transform," *IEEE Transactions on Nanabioscience,* vol. 2, no. 4, 184–189.
65. Ivana Duskumovic, Aleksandra Pizurica, Gjenna Stippel, Wilfried Philips, Ignace Lemahieu, 2000, "Wavelet Based Denoising Techniques for Ultrasound Images," *Proc. of the World Congress on Medical Physics and Biomedical Engineering,* Chicago, USA.
66. Qi Li, Tao Li, Shenghuo Zhu, Chandra Kambhamettu, 2002, "How well can wavelet denoising improve the accuracy of computing fundamental matrices?", *Proc. IEEE workshop on Motion and Video Computing*, 247–252.
67. Saleeb, A.F., Tseng, S.M., and Gendy, A.S., 1997, "Development and Evaluation Studies on New NDE Techniques for Damage Detection in Structures." Reprint.
68. Tseng, S.M., 1993, "Studies on Global Methods for Localized – Damage Detection in Large Scale Structures," The University of Akron, Ph.D. dissertation.
69. Ratnasumritkul, S., 1997, "Experimental Investigation of a New Global Damage Detection scheme," The University of Akron, Masters Thesis.
70. Gobin, P.F., Jayet, Y., and Baboux, J.C., et al., 2000, "New Trends in Non-Destructive Evaluation in Relation on the Smart Material Concept," *International Journal of Systems Science*, vol. 31, no. 11, 1351–1359.
71. Au, F.T.K., Cheng, Y.S., Tham, L.G., and Bai, Z.Z., 2003, "Structural Damage Detection Based on a Micro-Genetic Algorithm using Incomplete and Noisy Modal Test Data," *Journal of Sound and Vibration*, vol. 259, no. 5, 1081–1094.
72. Jonathan W. Pillow, and Eero P. Simoncelli, 2003, "Biases in White Noise Analysis due to Non-Poisson Spike Generation," *Journal of Neurocomputing*, vol. 52, no. 54, 109–115.
73. Juan Manuel Morales, 1999, "Viability in a Pink Environment: Why "White Noise Models can be Dangerous," *Ecology Letters*, vol. 2, 228–232.
74. Alin Achim, and Panagiotis Tsakalides, and Anastasios Bezerianos, 2003, "SAR Image Denoising via Bayesian Wavelet Shrinkage Based on Heavy-Tailed Modeling," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 8, 1773–1784.
75. Pierre Mouline, and Juan Liu, 1999, "Analysis of Multiresolution Image Denoising Schemes Using Generalized Gaussian and Complexity Priors," *IEEE Trasactions on Information Theory*, vol. 45, no. 3, 909–919.

76. Manfred Hertwig, Torsten Flemming, and Ralf Usinger, 1994, "Speckle Interferometry for Detection of Subsurface Damage in Fibre-Reinforced Composites," *Journal of Measurement Science and Technology*, vol. 5, 100–104.
77. Mindlin, D., 1951, "Influence of Rotary Inertia and Shear on Flexural Motions of Isotropic Elastic Plates," *Journal of Applied Mechanics*, 31–38.
78. Saleeb, A.F., and Gendy, A.S., 2000, "Nonlinear Dynamics for Mixed Shells with Large Rotation and Elasto-Plasticity," *International Journal of Computational Engineering Science*, vol. 1, no. 1, 1–31.
79. Saleeb, A.F., and Chang, T.Y., 1987, "An Efficient Quadrilateral Element for Plate Bending Analysis," *International Journal for Numerical Methods in Engineering*, vol. 24, 1123–1155.
80. Saleeb, A.F., Chang, T.Y., and Graf, W., 1987, "A Quadrilateral Shell Element using a Mixed Formulation," *Journal of Computer and Structures*, vol. 26, no. 5, 787–803.
81. Timoshenko, S.P., and Gere, J.M., 1972, "Mechanics of Materials," D. Van Nostrand Reinhold Company, New York.
82. Hinton, E., Salonen, E.M., and Bicanic, N., 1978, "A Study of Locking Phenomena in Isoparametric Elements," 3rd MAFELAP Conference, Brunel University, Uxbridge, UK.
83. Walker, J., 1999, "A Primer on Wavelets and Their Scientific Applications," Chapman and Hall Publishing.
84. Abbate, A., DeCusatis, C.M., and Das, P.K., 2002, "Wavelets and Subbands: Fundamentals and Applications," Birkhauser Publishing.
85. Milind, L. Prabhu, 2002, "Defect Localization Capabilities of a Global Damage Detection Scheme: Spatial Pattern Recognition using Full-Field Vibration Test Data in Plates," The University of Akron, Masters Thesis.

Appendix A
C++ Programs for Global Damage Detection

A.1 Program to Add the Noise to the Displacements in the Data Input File

```
/*********************************************************************/
//////////PROGRAM TO ADD THE NOISE TO THE DATA INPUT (.din) FILE ////////
/*********************************************************************/
#include<iostream>
#include<iomanip>
#include<cstdlib>
#include<fstream>
#include<math.h>

using namespace std;

//int main()
int main(int argc ,char * argv[])
{
    if(argc != 5)
    {
    cout<<"Error  ";
        cout<<"Usage: commandline< input_file noise1_file noise1_file        result_file >"<<endl;
    return -1;
    }

    char buffer[2093];
    int intTmp;
    int max_node;
    int max_element;
    double m,m1,m2;
//  int count=0;

    //ifstream inFile_din("phase1_format.din",ios::in);

    ifstream inFile_din(argv[1],ios::in);

    if(!inFile_din)
    {
            cerr<<".din file not found!!!!!!!"<<endl;
    }

    inFile_din.getline(buffer,2093);
    inFile_din>>intTmp;
    inFile_din>>max_node;
    inFile_din.getline(buffer,2093);
    inFile_din.getline(buffer,2093);
    inFile_din.getline(buffer,2093);
```

```
while(1)
{
        inFile_din>>intTmp;
        inFile_din.getline(buffer,2093);
        inFile_din.getline(buffer,2093);
        if(intTmp == max_node)
                break;
}

while(1)
{
        inFile_din>>intTmp;
        inFile_din.getline(buffer,2093);

        if(intTmp == max_node)
                break;
}

inFile_din.getline(buffer,2093);
inFile_din>>intTmp;
inFile_din>>max_element;

int mat;
inFile_din>>mat;
inFile_din.getline(buffer,2093);

for(int i=0; i<mat; i++)
{
  inFile_din.getline(buffer,2093);

 }

while(1)
{
        inFile_din>>intTmp;
        inFile_din.getline(buffer,2093);
        inFile_din.getline(buffer,2093);

        if(intTmp == max_element)
                break;
}
inFile_din.getline(buffer,2093);


/*****************************************************************/
//              Calculationg Norm of UnDamaged Displacements
/*****************************************************************/

float Unorm1, Unorm2, Unorm3, Unorm4, Unorm5, maxUnorm = 0.0;
Unorm1 = Unorm2 = Unorm3 = Unorm4 = Unorm5 = 0.0;
```

```
bool done = false;
int x=0;
double U1[10000],U2[10000],U3[10000],U4[10000],U5[10000];

while(!done)
{
        inFile_din>>intTmp;
        if(intTmp == max_node)
                done = true;

        inFile_din>>intTmp;
        inFile_din>>intTmp;

        inFile_din>>m1;
        U1[x]=m1;   //storing the undamaged x-displacements
        Unorm1 = Unorm1+m1*m1;

        inFile_din>>intTmp;
        inFile_din>>m1;
        U2[x]=m1;   //storing the undamaged y-displacements
        Unorm2 = Unorm2+m1*m1;

        inFile_din>>intTmp;
        inFile_din>>m1;
        U3[x]=m1;   //storing the undamaged z-displacements
Unorm3 = Unorm3+m1*m1;

        inFile_din>>intTmp;
        inFile_din>>m1;
        U4[x]=m1;   //storing the undamaged Rx-displacements
Unorm4 = Unorm4+m1*m1;

        inFile_din>>intTmp;
        inFile_din>>m1;
        U5[x]=m1;   //storing the undamaged Ry-displacements
Unorm5 = Unorm5+m1*m1;

        x++;
}

double udSignal1 = Unorm1;
double udSignal2 = Unorm2;
double udSignal3 = Unorm3;
double udSignal4 = Unorm4;
double udSignal5 = Unorm5;
double udSignal;

udSignal = Unorm1+Unorm2+Unorm3+Unorm4+Unorm5;

Unorm1 = sqrt(Unorm1);
```

```
        Unorm2 = sqrt(Unorm2);
        Unorm3 = sqrt(Unorm3);
        Unorm4 = sqrt(Unorm4);
        Unorm5 = sqrt(Unorm5);

        inFile_din.getline(buffer,2093);
        inFile_din.getline(buffer,2093);
        inFile_din.getline(buffer,2093);
/*****************************************************************/
//   Calculationg Norm of Damaged Displacements and Norm of Delta
/*****************************************************************/

float DeltaNorm1,Dnorm1,Dratio1,Uratio1,DeltaNorm2,Dnorm2,Dratio2,Uratio2,
       DeltaNorm3,Dnorm3,Dratio3,Uratio3, DeltaNorm4,Dnorm4,Dratio4,Uratio4,
       DeltaNorm5,Dnorm5,Dratio5,Uratio5, maxDnorm = 0.0;

        double D1[10000],D2[10000],D3[10000],D4[10000],D5[10000];
        double Delta;

        done = false;
        x=0;

        DeltaNorm1=Dnorm1 = 0.0;
        DeltaNorm2=Dnorm2 = 0.0;
        DeltaNorm3=Dnorm3 = 0.0;
        DeltaNorm4=Dnorm4 = 0.0;
        DeltaNorm5=Dnorm5 = 0.0;
        while(!done)
        {
                inFile_din>>intTmp;
                if(intTmp == max_node)
                        done = true;

                inFile_din>>intTmp;
                inFile_din>>intTmp;

                inFile_din>>m1;
                Dnorm1 = Dnorm1+ m1*m1;
                D1[x] = m1;
                Delta=D1[x]-U1[x];      //(Damage - undamage = Delta)
                DeltaNorm1 = DeltaNorm1+Delta*Delta;

                inFile_din>>intTmp;
                inFile_din>>m1;
                Dnorm2 = Dnorm2+ m1*m1;
                D2[x] = m1;
                Delta=D2[x]-U2[x];      //(Damage - undamage = Delta)
                DeltaNorm2 = DeltaNorm2+Delta*Delta;

                inFile_din>>intTmp;
                inFile_din>>m1;
```

```
            Dnorm3 = Dnorm3+ m1*m1;
            D3[x] = m1;
            Delta=D3[x]-U3[x];       //(Damage - undamage = Delta)
            DeltaNorm3 = DeltaNorm3+Delta*Delta;

            inFile_din>>intTmp;
            inFile_din>>m1;
            Dnorm4 = Dnorm4+ m1*m1;
            D4[x] = m1;
            Delta=D4[x] - U4[x];       //(Damage - undamage = Delta)
            DeltaNorm4 = DeltaNorm4+Delta*Delta;

            inFile_din>>intTmp;
            inFile_din>>m1;
            Dnorm5 = Dnorm5+ m1*m1;
            D5[x] = m1;
            Delta=D5[x]- U5[x];       //(Damage - undamage = Delta)
            DeltaNorm5 = DeltaNorm5+Delta*Delta;

            x++;
}
double dSignal1 = Dnorm1;
double dSignal2 = Dnorm2;
double dSignal3 = Dnorm3;
double dSignal4 = Dnorm4;
double dSignal5 = Dnorm5;
double dSignal;

dSignal = Dnorm1+Dnorm2+Dnorm3+Dnorm4+Dnorm5;

Dnorm1 = sqrt(Dnorm1);
Dnorm2 = sqrt(Dnorm2);
Dnorm3 = sqrt(Dnorm3);
Dnorm4 = sqrt(Dnorm4);
Dnorm5 = sqrt(Dnorm5);

DeltaNorm1 = sqrt(DeltaNorm1);
DeltaNorm2 = sqrt(DeltaNorm2);
DeltaNorm3 = sqrt(DeltaNorm3);
DeltaNorm4 = sqrt(DeltaNorm4);
DeltaNorm5 = sqrt(DeltaNorm5);

if(Unorm1==0) Uratio1 =0;
else Uratio1 = DeltaNorm1/Unorm1;

 if(Unorm2==0) Uratio2 =0;
else Uratio2 = DeltaNorm2/Unorm2;

if(Unorm3==0) Uratio3 =0;
else Uratio3 = DeltaNorm3/Unorm3;
```

```
if(Unorm4==0) Uratio4 =0;
else Uratio4 = DeltaNorm4/Unorm4;

if(Unorm5==0) Uratio5 =0;
else Uratio5 = DeltaNorm5/Unorm5;

if(Dnorm1==0) Dratio1 =0;
else Dratio1 = DeltaNorm1/Dnorm1;

if(Dnorm2==0) Dratio2 =0;
else Dratio2 = DeltaNorm2/Dnorm2;

if(Dnorm3==0) Dratio3 =0;
else Dratio3 = DeltaNorm3/Dnorm3;

if(Dnorm4==0) Dratio4 =0;
else Dratio4 = DeltaNorm4/Dnorm4;

if(Dnorm5==0) Dratio5 =0;
else Dratio5 = DeltaNorm5/Dnorm5;

if(Unorm1 > maxUnorm)
                maxUnorm = Unorm1;
        else if(Unorm2 > maxUnorm)
                maxUnorm = Unorm2;
        else if(Unorm3 > maxUnorm)
                maxUnorm = Unorm3;
        else if(Unorm4 > maxUnorm)
                maxUnorm = Unorm4;
        else if(Unorm5 > maxUnorm)
                maxUnorm = Unorm5;


if(Dnorm1 > maxDnorm)
                maxDnorm = Dnorm1;
        else if(Dnorm2 > maxDnorm)
                maxDnorm = Dnorm2;
        else if(Dnorm3 > maxDnorm)
                maxDnorm = Dnorm3;
        else if(Dnorm4 > maxDnorm)
                maxDnorm = Dnorm4;
        else if(Dnorm5 > maxDnorm)
                maxDnorm = Dnorm5;

inFile_din.getline(buffer,2093);
inFile_din.getline(buffer,2093);
inFile_din.getline(buffer,2093);
inFile_din.getline(buffer,2093);
inFile_din.getline(buffer,2093);

inFile_din.close();
```

```
/*************************************************************************/
//         End of all calculations and input file is closed
/*************************************************************************/

//   Opening the input file again to add the Noise
      //ifstream inFile_Noise("NormalizedWhiteNoise.dat",ios::in);
       inFile_din.open(argv[1],ios::in);


      //ofstream outFile_din("phase1_plus_noise.din",ios::out);
      ofstream outFile_din(argv[4],ios::out);

      inFile_din.getline(buffer,2093);
      outFile_din<<setw(5)<<buffer<<endl;

      inFile_din>>intTmp;
      outFile_din<<intTmp;

      inFile_din>>max_node;
      outFile_din<<setw(5)<<max_node;

      inFile_din.getline(buffer,2093);
      outFile_din<<buffer<<endl;

      inFile_din.getline(buffer,2093);
      outFile_din<<buffer<<endl;

      inFile_din.getline(buffer,2093);
      outFile_din<<buffer<<endl;

      while(1)
      {
               inFile_din>>intTmp;
               outFile_din<<setw(5)<<intTmp;

               inFile_din.getline(buffer,2093);
               outFile_din<<buffer<<endl;

               inFile_din.getline(buffer,2093);
               outFile_din<<buffer<<endl;
               if(intTmp == max_node)
                        break;
      }
      while(1)
      {
               inFile_din>>intTmp;
               outFile_din<<setw(5)<<intTmp;

               inFile_din.getline(buffer,2093);
               outFile_din<<buffer<<endl;
```

```
            if(intTmp == max_node)
                break;
        }
        inFile_din.getline(buffer,2093);
        outFile_din<<setw(5)<<buffer<<endl;

        inFile_din>>intTmp;
        outFile_din<<intTmp;

        inFile_din>>max_element;
        outFile_din<<setw(5)<<max_element;

//      int mat;
        inFile_din>>mat;
        outFile_din<<setw(5)<<mat;

        inFile_din.getline(buffer,2093);
        outFile_din<<buffer<<endl;

        for(i=0; i<mat; i++)
        {
            inFile_din.getline(buffer,2093);
            outFile_din<<buffer<<endl;
         }
        while(1)
        {
            inFile_din>>intTmp;
            outFile_din<<setw(8)<<intTmp;
            inFile_din.getline(buffer,2093);
            outFile_din<<buffer<<endl;
            inFile_din.getline(buffer,2093);
            outFile_din<<buffer<<endl;

            if(intTmp == max_element)
                break;
        }
        inFile_din.getline(buffer,2093);
        outFile_din<<setw(5)<<buffer<<endl;



    /*********************************************************************/
    //      Adding Noise to the UnDamaged Displacements
    /*********************************************************************/
        ifstream inFile_Noise(argv[2],ios::in);
        if(!inFile_Noise)
        {
            cerr<<"noise file not found!!!!!!!"<<endl;
            return -1;
        }
        done = false;
```

```
x=0;

double udNoise1 = 0.00;
double udNoise2 = 0.00;
double udNoise3 = 0.00;
double udNoise4 = 0.00;
double udNoise5 = 0.00;
double udNoise  = 0.00;
double udSNR, udSNR1, udSNR2, udSNR3, udSNR4, udSNR5;

while(!done)
{
        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        if(intTmp == max_node)
                done = true;

        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise>>m2;

        udNoise = udNoise + m2*m2;
        m= m1*(1+m2*Uratio1*Unorm1/maxUnorm);
outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

        udNoise1 = udNoise1 + pow(m1*m2*Uratio1*Unorm1/maxUnorm,2);
        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise>>m2;

        udNoise = udNoise + m2*m2;
             m= m1*(1+m2*Uratio2*Unorm2/maxUnorm);
        outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

        udNoise2 = udNoise2 + pow(m1*m2*Uratio2*Unorm2/maxUnorm,2);

        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise>>m2;

        udNoise = udNoise + m2*m2;
```

```
            m= m1*(1+m2*Uratio3*Unorm3/maxUnorm);
        outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

        udNoise3 = udNoise3 + pow(m1*m2*Uratio3*Unorm3/maxUnorm,2);

        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise>>m2;

        udNoise = udNoise + m2*m2;

        m= m1*(1+m2*Uratio4*Unorm4/maxUnorm);
                outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

        udNoise4 = udNoise4 + pow(m1*m2*Uratio4*Unorm4/maxUnorm,2);
        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise>>m2;

        udNoise = udNoise + m2*m2;
            m= m1*(1+m2*Uratio5*Unorm5/maxUnorm);
        outFile_din<<setw(25)<<setprecision(16)<<fixed<<m<<endl;

        udNoise5 = udNoise5 + pow(m1*m2*Uratio5*Unorm5/maxUnorm,2);
        x++;
    }

    udSNR1 = 10*log10(udSignal1/udNoise1);
    udSNR2 = 10*log10(udSignal2/udNoise2);
    udSNR3 = 10*log10(udSignal3/udNoise3);
    udSNR4 = 10*log10(udSignal4/udNoise4);
    udSNR5 = 10*log10(udSignal5/udNoise5);
    udSNR = 10*log10(udSignal/udNoise);
    inFile_din.getline(buffer,2093);
    outFile_din<<buffer;

    inFile_din.getline(buffer,2093);
    outFile_din<<setw(1)<<buffer<<endl;

    inFile_din.getline(buffer,2093);
    outFile_din<<setw(5)<<buffer<<endl;

/*****************************************************************/
//          Adding Noise to the Damaged Displacements
/*****************************************************************/
```

```
//ifstream inFile_Noise2("NormalizedPinkNoise.dat",ios::in);
ifstream inFile_Noise2(argv[3],ios::in);

if(!inFile_Noise2)
{
        cerr<<"noise.dat file not found!!!!!!!"<<endl;
        return -1;
}

done = false;
x=0;

double dNoise1 = 0.00;
double dNoise2 = 0.00;
double dNoise3 = 0.00;
double dNoise4 = 0.00;
double dNoise5 = 0.00;
double dNoise  = 0.00;
double dSNR, dSNR1, dSNR2, dSNR3, dSNR4, dSNR5;

while(!done)
{
        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        if(intTmp == max_node)
                done = true;

        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise2>>m2;

        dNoise = dNoise + m2*m2;
        m= m1*(1+m2*Dratio1*Dnorm1/maxDnorm);
outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

        dNoise1 = dNoise1 + pow(m1*m2*Dratio1*Dnorm1/maxUnorm,2);
        inFile_din>>intTmp;
        outFile_din<<setw(5)<<intTmp;

        inFile_din>>m1;
        inFile_Noise2>>m2;

        dNoise = dNoise + m2*m2;
             m= m1*(1+m2*Dratio2*Dnorm2/maxDnorm);
        outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;
```

```
            dNoise2 = dNoise2 + pow(m1*m2*Dratio2*Dnorm2/maxUnorm,2);

            inFile_din>>intTmp;
            outFile_din<<setw(5)<<intTmp;

            inFile_din>>m1;
            inFile_Noise2>>m2;

            dNoise = dNoise + m2*m2;
                m= m1*(1+m2*Dratio3*Dnorm3/maxDnorm);
            outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

            dNoise3 = dNoise3 + pow(m1*m2*Dratio3*Dnorm3/maxUnorm,2);
            inFile_din>>intTmp;
            outFile_din<<setw(5)<<intTmp;

            inFile_din>>m1;
            inFile_Noise2>>m2;

            dNoise = dNoise + m2*m2;
                m= m1*(1+m2*Dratio4*Dnorm4/maxDnorm);
            outFile_din<<setw(25)<<setprecision(16)<<fixed<<m;

            dNoise4 = dNoise4 + pow(m1*m2*Dratio4*Dnorm4/maxUnorm,2);

            inFile_din>>intTmp;
            outFile_din<<setw(5)<<intTmp;

            inFile_din>>m1;
            inFile_Noise2>>m2;

            dNoise = dNoise + m2*m2;
                m= m1*(1+m2*Dratio5*Dnorm5/maxDnorm);
            outFile_din<<setw(25)<<setprecision(16)<<fixed<<m<<endl;

            dNoise5 = dNoise5 + pow(m1*m2*Dratio5*Dnorm5/maxUnorm,2);
            x++;
    }

    dSNR1 = 10*log10(dSignal1/dNoise1);
    dSNR2 = 10*log10(dSignal2/dNoise2);
    dSNR3 = 10*log10(dSignal3/dNoise3);
    dSNR4 = 10*log10(dSignal4/dNoise4);
    dSNR5 = 10*log10(dSignal5/dNoise5);

    dSNR = 10*log10(dSignal/dNoise);

    double SNR, Signal, Noise;

    Signal = udSignal+dSignal;
```

```
    Noise = udNoise+dNoise;

    SNR = 10*log10(Signal/Noise);
    cout<<"\nDISPLACEMENTS TO NOISE RATIO (SNR):";
    cout<<setw(10)<<SNR<<endl;
/*****************************************************************/

    cout<<"\nAVERAGE MAGNITUDE (RMS VALUE) OF "
          <<"THE NOISE ADDED TO THE DISPLACEMENTS: ";
    cout<<sqrt(Noise/(1681*5))<<endl;

    inFile_din.getline(buffer,2093);
    outFile_din<<buffer;

    inFile_din.getline(buffer,2093);
    outFile_din<<buffer<<endl;

    inFile_din.getline(buffer,2093);
    outFile_din<<buffer<<endl;

    inFile_din.getline(buffer,2093);
    outFile_din<<buffer<<endl;

    inFile_din.getline(buffer,2093);
    outFile_din<<buffer<<endl;

    inFile_Noise.close();
    inFile_Noise2.close();
    inFile_din.close();
    outFile_din.close();

    return 0;
}
```

A.2 Generation of White Noise and Pink Noise

```
/*****************************************************************/
//        GENERATION OF WHITE NOISE AND PINK NOISE
/*****************************************************************/

#include<iostream>
#include<iomanip>
#include<cstdlib>
#include<ctime>
#include<fstream>

using namespace std;

//int main()
```

```
int main(int argc ,char * argv[])
{
    if(argc !=3 )
    {
    cout<<"Error  ";
    cout<<"Usage: commandline< White_Noise_file Pink_Noise_file >"<<endl;
    return -1;
    }

// Dynamic memory allocation for an Array
    float *white_noise1;
    int SizeOfData;
    cout<<"\nENTER THE NUMBER OF NODAL DISPLACEMENTS:";
    cin>>SizeOfData;

    white_noise1 = new float[SizeOfData];
//End of allocation process

    float *white_noise2;
    white_noise2 = new float[SizeOfData];

    float *white_noise3;
    white_noise3 = new float[SizeOfData];

    float *white_noise4;
    white_noise4 = new float[SizeOfData];

    float *white_noise5;
    white_noise5 = new float[SizeOfData];

/*********************************************************************/
//      Generating the white noise using rand() function
/*********************************************************************/

    srand(time (0));

    for( int i=0; i<SizeOfData; i++)
    {
            white_noise1[i] = rand();
            white_noise2[i] = rand();
            white_noise3[i] = rand();
            white_noise4[i] = rand();
            white_noise5[i] = rand();

/*********************************************************************/
//              NORMALIZING THE WHITE NOISE
/*********************************************************************/
    int min1, max1 = 0;
    int min2, max2 = 0;
    int min3, max3 = 0;
    int min4, max4 = 0;
```

```
int min5, max5 = 0;

//float normalized_white_noise[SizeOfData];
float *normalized_white_noise1;
normalized_white_noise1 = new float[SizeOfData];

float *normalized_white_noise2;
normalized_white_noise2 = new float[SizeOfData];

float *normalized_white_noise3;
normalized_white_noise3 = new float[SizeOfData];

float *normalized_white_noise4;
normalized_white_noise4 = new float[SizeOfData];

float *normalized_white_noise5;
normalized_white_noise5 = new float[SizeOfData];

for(i=0; i<SizeOfData; i++)
{
        if(white_noise1[i] > max1)
                max1 = white_noise1[i];

        if(white_noise2[i] > max2)
                max2 = white_noise2[i];

        if(white_noise3[i] > max3)
                max3 = white_noise3[i];

        if(white_noise4[i] > max4)
                max4 = white_noise4[i];

        if(white_noise5[i] > max5)
                max5 = white_noise5[i];
}
min1 = max1;
min2 = max2;
min3 = max3;
min4 = max4;
min5 = max5;

for(i=0; i<SizeOfData; i++)
{
        if(white_noise1[i] < min1)
                min1 = white_noise1[i];

        if(white_noise2[i] < min2)
                min2 = white_noise2[i];

        if(white_noise3[i] < min3)
                min3 = white_noise3[i];
```

```
            if(white_noise4[i] < min4)
                    min4 = white_noise4[i];

            if(white_noise5[i] < min5)
                    min5 = white_noise5[i];
      }

      //Opening the same buffer file to create another file
      //outFile.open("NormalizedWhiteNoise.dat",ios::out);

      ofstream outFile1(argv[1],ios::out);
      float factor1,factor2,factor3,factor4,factor5;
      float percent1,percent2,percent3,percent4,percent5;


      float factor;
      float percent;
      int mid1, mid2, mid3, mid4, mid5;

      mid1 = (min1+max1)/2;
      mid2 = (min2+max2)/2;
      mid3 = (min3+max3)/2;
      mid4 = (min4+max4)/2;
      mid5 = (min5+max5)/2;


   /* cin>>percent;
      percent = percent/2;
      factor=percent/100;*/

      cout<<"\nENTER THE PERCENTAGE OF THE NOISE LEVEL IN X-
DISPLACEMENTS: ";
      cin>>percent1;

      cout<<"\nENTER THE PERCENTAGE OF THE NOISE LEVEL IN Y-
DISPLACEMENTS: ";
      cin>>percent2;

      cout<<"\nENTER THE PERCENTAGE OF THE NOISE LEVEL IN Z-
DISPLACEMENTS: ";
      cin>>percent3;

      cout<<"\nENTER THE PERCENTAGE OF THE NOISE LEVEL IN R1-ROTATIONS:
";
      cin>>percent4;

      cout<<"\nENTER THE PERCENTAGE OF THE NOISE LEVEL IN R2-ROTATIONS:
";
      cin>>percent5;

      factor1=percent1/200;
```

```
        factor2=percent2/200;
        factor3=percent3/200;
        factor4=percent4/200;
        factor5=percent5/200;

        for(i=0; i<SizeOfData; i++)
        {
                if(white_noise1[i]<=mid1)
                  normalized_white_noise1[i] = (white_noise1[i] - mid1)/(mid1-min1);
                else
                  normalized_white_noise1[i] = (white_noise1[i] - mid1)/(max1-mid1);


                if(white_noise2[i]<=mid2)
                  normalized_white_noise2[i] = (white_noise2[i] - mid2)/(mid2-min2);
                else
                  normalized_white_noise2[i] = (white_noise2[i] - mid2)/(max2-mid2);

                if(white_noise3[i]<=mid3)
                  normalized_white_noise3[i] = (white_noise3[i] - mid3)/(mid3-min3);
                else
                  normalized_white_noise3[i] = (white_noise3[i] - mid3)/(max3-mid3);


                if(white_noise4[i]<=mid4)
                  normalized_white_noise4[i] = (white_noise4[i] - mid4)/(mid4-min4);
                else
                  normalized_white_noise4[i] = (white_noise4[i] - mid4)/(max4-mid4);


                if(white_noise5[i]<=mid5)
                  normalized_white_noise5[i] = (white_noise5[i] - mid5)/(mid5-min5);
                else
                  normalized_white_noise5[i] = (white_noise5[i] - mid5)/(max5-mid5);

                outFile1<<setw(20)<<setprecision(16)<<fixed<<normalized_white_noise1[i]*factor1<<"
"

<<setw(20)<<setprecision(16)<<fixed<<normalized_white_noise2[i]*factor2<<"    "

<<setw(20)<<setprecision(16)<<fixed<<normalized_white_noise3[i]*factor3<<"    "

<<setw(20)<<setprecision(16)<<fixed<<normalized_white_noise4[i]*factor4<<"    "

<<setw(20)<<setprecision(16)<<fixed<<normalized_white_noise5[i]*factor5<<"    "
                                <<endl;
        }

        outFile1.close(); //closing the file
```

```
/*************************************************************************/
//            Filtering the White Noise to generate the Pink Noise
/*************************************************************************/
//Assuming that:
//data is 8 bits
//sample rate for data is 44 KHz and
//low pass F is set at 5.5 KHz
//Therefore 44/5.5 = 8

    #define F 8

    int  avg1, avg2, avg3, avg4, avg5;
    int  cnt;
    //float pink_noise[SizeOfData];

    float *pink_noise1;
    pink_noise1 = new float[SizeOfData];
    float *pink_noise2;
    pink_noise2 = new float[SizeOfData];

    float *pink_noise3;
    pink_noise3 = new float[SizeOfData];

    float *pink_noise4;
    pink_noise4 = new float[SizeOfData];

    float *pink_noise5;
    pink_noise5 = new float[SizeOfData];

    //Opening the same buffer file to create another file
    //outFile.open("pinknoise.dat",ios::out);

    for (i = 0; i < SizeOfData; i++)
    {
            for (avg1 = 0, avg2 = 0, avg3 = 0, avg4 = 0, avg5 = 0, cnt = 0;
            cnt < F && (i + cnt) < SizeOfData; cnt++)
            {
                    avg1 += white_noise1[i + cnt];  //average of several samples
                    avg2 += white_noise2[i + cnt];
                    avg3 += white_noise3[i + cnt];
                    avg4 += white_noise4[i + cnt];
                    avg5 += white_noise5[i + cnt];
            }

            pink_noise1[i] = avg1/cnt;  //generation of pink noise
       pink_noise2[i] = avg2/cnt;
            pink_noise3[i] = avg3/cnt;
            pink_noise4[i] = avg4/cnt;
            pink_noise5[i] = avg5/cnt;

    }
```

```
/*********************************************************************/
//                    NORMALIZING THE PINK NOISE
/*********************************************************************/

    //float normalized_pink_noise[SizeOfData];
    float *normalized_pink_noise1;
    normalized_pink_noise1 = new float[SizeOfData];

    float *normalized_pink_noise2;
    normalized_pink_noise2 = new float[SizeOfData];
    float *normalized_pink_noise3;
    normalized_pink_noise3 = new float[SizeOfData];

    float *normalized_pink_noise4;
    normalized_pink_noise4 = new float[SizeOfData];

    float *normalized_pink_noise5;
    normalized_pink_noise5 = new float[SizeOfData];

    min1, max1 = 0;
    min2, max2 = 0;
    min3, max3 = 0;
    min4, max4 = 0;
    min5, max5 = 0;

    for(i=0; i<SizeOfData; i++)
    {
            if(pink_noise1[i] > max1)
                    max1 = pink_noise1[i];

            if(pink_noise2[i] > max2)
                    max2 = pink_noise2[i];

            if(pink_noise3[i] > max3)
                    max3 = pink_noise3[i];

            if(pink_noise4[i] > max4)
                    max4 = pink_noise4[i];

            if(pink_noise5[i] > max5)
                    max5 = pink_noise5[i];
    }
    min1 = max1;
    min2 = max2;
    min3 = max3;
    min4 = max4;
    min5 = max5;

    for(i=0; i<SizeOfData; i++)
    {
```

```
		if(pink_noise1[i] < min1)
			min1 = pink_noise1[i];

		if(pink_noise2[i] < min2)
			min2 = pink_noise2[i];
		if(pink_noise3[i] < min3)
			min3 = pink_noise3[i];

		if(pink_noise4[i] < min4)
			min4 = pink_noise4[i];

		if(pink_noise5[i] < min5)
			min5 = pink_noise5[i];
}
mid1 = (min1+max1)/2;
mid2 = (min2+max2)/2;
mid3 = (min3+max3)/2;
mid4 = (min4+max4)/2;
mid5 = (min5+max5)/2;

//Opening the same buffer file to create another file
//outFile.open("NormalizedPinkNoise.dat",ios::out);

ofstream outFile2(argv[2],ios::out);
for(i=0; i<SizeOfData; i++)
{
		if(pink_noise1[1]<= mid1)
		  normalized_pink_noise1[i] = (pink_noise1[i] - mid1)/(mid1-min1);
		else
		  normalized_pink_noise1[i] = (pink_noise1[i] - mid1)/(max1-mid1);

		if(pink_noise2[1]<= mid2)
		  normalized_pink_noise2[i] = (pink_noise2[i] - mid2)/(mid2-min2);
		else
		  normalized_pink_noise2[i] = (pink_noise2[i] - mid2)/(max2-mid2);

		if(pink_noise3[1]<= mid3)
		  normalized_pink_noise3[i] = (pink_noise3[i] - mid3)/(mid3-min3);
		else
		  normalized_pink_noise3[i] = (pink_noise3[i] - mid3)/(max3-mid3);

		if(pink_noise4[1]<= mid4)
		  normalized_pink_noise4[i] = (pink_noise4[i] - mid4)/(mid4-min4);
		else
		  normalized_pink_noise4[i] = (pink_noise4[i] - mid4)/(max4-mid4);

		if(pink_noise5[1]<= mid5)
		  normalized_pink_noise5[i] = (pink_noise5[i] - mid5)/(mid5-min5);
		else
		  normalized_pink_noise5[i] = (pink_noise5[i] - mid5)/(max5-mid5);
outFile2<<setw(20)<<setprecision(16)<<fixed<<normalized_pink_noise1[i]*factor1<<"    "
```

```
<<setw(20)<<setprecision(16)<<fixed<<normalized_pink_noise2[i]*factor2<<"    "

<<setw(20)<<setprecision(16)<<fixed<<normalized_pink_noise3[i]*factor3<<"    "

<<setw(20)<<setprecision(16)<<fixed<<normalized_pink_noise4[i]*factor4<<"    "

<<setw(20)<<setprecision(16)<<fixed<<normalized_pink_noise5[i]*factor5<<"    "
                                <<endl;
        }

        outFile2.close();  //closing the file

        delete [] white_noise1;
        delete [] white_noise2;
        delete [] white_noise3;
        delete [] white_noise4;
        delete [] white_noise5;

        delete [] normalized_white_noise1;
        delete [] normalized_white_noise2;
        delete [] normalized_white_noise3;
        delete [] normalized_white_noise4;
        delete [] normalized_white_noise5;

        delete [] pink_noise1;
        delete [] pink_noise2;
        delete [] pink_noise3;
        delete [] pink_noise4;
        delete [] pink_noise5;

        delete [] normalized_pink_noise1;
        delete [] normalized_pink_noise2;
        delete [] normalized_pink_noise3;
        delete [] normalized_pink_noise4;
        delete [] normalized_pink_noise5;

        return 0;
    }
```

A.3 Generation of Noised Data Output File and Noised Damage Parameter Signal and also Denoised Data Output File Which is Processed using Matlab

```
/*****************************************************************/
//                GENERATION OF NOISED DATA OUTPUT FILE and      //
//     DENOISED OUTPUT FILE WHICH IS PROCESSED USING MATLAB      //
/*****************************************************************/
```

```
//DESCRIPTION
//This program adds the noise to the smooth damage parameter values of data output file
//and stores them in different file to import in Matlab as a signal.
//And also creates the denoised data output file after denoising the smooth damage //parameter values
using Matlab Wavelet Toolbox.

#include<iostream>
#include<iomanip>
#include<cstdlib>
#include<fstream>
#include<math.h>
#include<ctime>

using namespace std;

//int main()
int main(int argc ,char * argv[])
{
    int ele, intmp, choice;
    double dtmp, noise, percent, factor, NSDP;
    char buffer[2093];
    char check1[] = "D A M A G E   P A R A M E T E R   F O R   M O D E =    1";
    double maxPar=0.0000;

    while(1)
    {
        cout<<"TO CREATE NOISED DATA OUTPUT FILE     :  1"<<endl<<endl
            <<"TO CREATE DENOISED DATA OUTPUT FILE   :  2"<<endl<<endl
            <<"ENTER YOUR CHOICE : ";
        cin>>choice;

        if(choice !=1 && choice != 2)
        {
            cout<<"\nENTER 1 OR 2 FOR YOUR CHOICE\n"<<endl;
            return -1;
        }
        else
            break;
    }

//CHECKING THE COMMAND LINE ARGUMENTS
    if(choice==1)
    {
        if(argc != 4)
        {
            cout<<"Error  ";
            cout<<"Usage: commandline< dotFile result_dotFile result_NSDPfile>"<<endl;
            return -1;
        }
    }
    else
```

```
	{
		if(argc != 4)
		{
			cout<<"Error  ";
			cout<<"Usage: commandline< dotFile Denoised_SDPfile
result_denoised_dotFile >"<<endl;
			return -1;
		}

	}
	cout<<"\nENTER THE NUMBER OF ELEMENTS = ";
	cin>>ele;


    //ADDING THE NOISE TO THE DAMAGE PARAMETER VALUES AND
    //CREATING NOISED DAMAGED PARAMETER FILE TO DENOISE USING THE //MATLAB
WAVELET TOOLBOX

	if(choice == 1)
	{
		//ifstream in_dot("test.dot", ios::in);
		ifstream in_dot(argv[1], ios::in);
		if(!in_dot)
		{
			cerr<<".dot file not found!!!!!!!"<<endl;
		}


		while(1)
		{
			in_dot.getline(buffer,2093);

			int ch = strcmp(buffer, check1);
			if(ch==0)
			{
				in_dot.getline(buffer,2093);
				in_dot.getline(buffer,2093);
				in_dot.getline(buffer,2093);
				in_dot.getline(buffer,2093);

				//RAW DAMAGE PARAMETER
				while(1)
				{
					in_dot>>intmp;
					in_dot>>dtmp;
					in_dot>>dtmp;
					in_dot>>dtmp;
					in_dot>>dtmp;
					in_dot>>dtmp;
					in_dot>>dtmp;
					if(intmp==ele)
```

```
                                        break;
                                }

                                in_dot.getline(buffer,2093);
                                in_dot.getline(buffer,2093);
                                in_dot.getline(buffer,2093);
                                in_dot.getline(buffer,2093);
                                in_dot.getline(buffer,2093);

        //SMOOTH DAMAGE PARAMETER....(Finding maximum in the values)
                                while(1)
                                        {
                                                in_dot>>intmp;
                                                in_dot>>dtmp;
                                                in_dot>>dtmp;
                                                in_dot>>dtmp;
                                                in_dot>>dtmp;
                                                in_dot>>dtmp;

                                                if(dtmp > maxPar)
                                                        maxPar = dtmp;

                                                if(intmp==ele)
                                                        break;
                                        }
                                break;
                        }
                }
                in_dot.close();
/*************************************************************************/
                in_dot.open(argv[1],ios::in);
                if(!in_dot)
                {
                        cerr<<".dot file not found!!!!!!!"<<endl;
                        return -1;
                }

                srand(time (0));
                cout<<"\nENTER THE PERCENTAGE OF THE NOISE LEVEL = ";
                cin>>percent;
//              percent = percent/2;
                factor = percent/100;

//              ofstream out_dot("noisetest.dot", ios::out);
                ofstream out_dot(argv[2], ios::out);
//              ofstream out_matlab("NSDPvectors.dat", ios::out);
                ofstream out_matlab(argv[3], ios::out);

                while(1)
                {
                        in_dot.getline(buffer,2093);
```

```
                out_dot<<buffer<<endl;

                int ch = strcmp(buffer, check1);
                if(ch==0)
                {
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;

                        //RAW DAMAGE PARAMETER
                        while(1)
                        {
                                in_dot>>intmp;
                                out_dot<<setw(6)<<intmp;

                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                //RANDOM NUMBER GENERATION
                                noise = 2*(0.5-((double)rand()/(double)(RAND_MAX+1)));
                                noise = maxPar*noise*factor;

                                in_dot>>dtmp;
                                out_dot<<setw(20)<<setprecision(12)<<fixed
                                        <<dtmp+noise<<endl;

                                if(intmp==ele)
                                        break;
                        }

                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
```

```
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;

                        while(1)
                                {
                                        in_dot>>intmp;
                                        out_dot<<setw(6)<<intmp;

                                        in_dot>>dtmp;
                                        out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
                                        out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
                                        out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
                                        out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        //RANDOM NUMBER GENERATION
                                        noise = 2*(0.5-
((double)rand()/(double)(RAND_MAX+1)));
                                        noise = maxPar*noise*factor;

                                        double ndtmp;

                                        in_dot>>dtmp;

                                        ndtmp = dtmp+noise;

                                        out_dot<<setw(20)<<setprecision(9)<<fixed
                                                <<ndtmp<<endl;
                                        out_matlab<<setw(20)<<setprecision(9)<<fixed
                                                <<ndtmp<<endl;

                                        if(intmp==ele)
                                                break;
                                }
                        break;
                }
        }

                while(1)
                {
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
```

```
                        if(in_dot.eof())
                                break;
                }

                in_dot.close();
                out_dot.close();
                out_matlab.close();
        }

    //CREATING THE DATA OUTPUT FILE WITH DENOISED DAMAGE //PARAMETER
VALUES

        if(choice == 2)
        {
    //          ifstream in_dot("test.dot", ios::in);
                ifstream in_dot(argv[1], ios::in);
                if(!in_dot)
                {
                        cerr<<".dot file not found!!!!!!!"<<endl;
                }

    //          ifstream in_matlab("NSDPvectors.dat", ios::in);
                ifstream in_matlab(argv[2], ios::in);
                if(!in_matlab)
                {
                        cerr<<"denoised file not found!!!!!!!"<<endl;
                }

    //          ofstream out_dot("Denoisetest.dot", ios::out);
                ofstream out_dot(argv[3], ios::out);

                while(1)
                {
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;

                        int ch = strcmp(buffer, check1);
                        if(ch==0)
                        {
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;

        //RAW DAMAGE PARAMETER .....(denoising only "smooth" for now)
                                while(1)
                                {
```

```
                                        in_dot>>intmp;
                                        out_dot<<setw(6)<<intmp;

                                        in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                        in_dot>>dtmp;
                                        out_dot<<setw(20)<<setprecision(12)<<fixed
                                                <<dtmp<<endl;

                                        if(intmp==ele)
                                                break;
                                }
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                in_dot.getline(buffer,2093);
                                out_dot<<buffer<<endl;
                                        while(1)
                                        {
                                                in_dot>>intmp;
                                                out_dot<<setw(6)<<intmp;

                                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;

                                                in_dot>>dtmp;
out_dot<<setw(20)<<setprecision(12)<<fixed<<dtmp;
```

```
                                        in_dot>>dtmp;
                                        in_matlab>>NSDP;

            out_dot<<setw(20)<<setprecision(12)<<fixed<<NSDP<<endl;

                                        if(intmp==ele)
                                                break;
                                }
                        break;
                }
        }
                while(1)
                {
                        in_dot.getline(buffer,2093);
                        out_dot<<buffer<<endl;
                        if(in_dot.eof())
                                break;
                }
        in_dot.close();
        out_dot.close();
        in_matlab.close();

    }
    return 0;
}
```

A.4 Generation of Data Input File using the displacements/mode shapes from the ABAQUS Modal Analysis

```
/*****************************************************************/
//////////GENERATION THE DATA INPUT FILE FOR THE DAMAGE CODE//////////
/////USING THE DISPLACEMENTS FROM THE ABAQUS MODAL ANALYSIS/////
/*****************************************************************/

//PASS THE UNDAMAGED AND DAMAGED DISPLACEMENTS FILE WITHOUT
//THE MINIMUM, MAXIMUM AND TOTAL AT THE END OF THE FILE

#include<iostream>
#include<iomanip>
#include<cstdlib>
#include<fstream>
#include<math.h>

using namespace std;

int getVectors(char s[], char o[], int Ex, int Ey);
```

```
//int main()
int main(int argc, char *argv[])
{
    if(argc != 4)
    {
    cout<<"Error  ";
    cout<<"Usage: commandline< Undamaged_disp_file Damaged_disp_file result_file >"<<endl;
    return -1;
    }

    int nodeID = 1;
    int max_node, ch;
    int max_element;
    double l, w, t;
    int Xele,Yele;

    ch = nodeID;

    cout<<"\nNOTE: Undamaged and Damaged displacements files shouldn't have"<<endl
        <<"       the minimum, maximum and total at the end of the file"<<endl<<endl;

    ofstream outFile_din(argv[3], ios::out);
//  ofstream outFile_din("temp.din", ios::out);

    cout<<"ENTER THE LENGTH(IN X-DIRECTION) OF THE PLATE = ";
    cin>>l;
    cout<<endl;

    cout<<"ENTER THE WIDTH (IN Y-DIRECTION) OF THE PLATE = ";
    cin>>w;
    cout<<endl;

    cout<<"ENTER THE THICKNESS OF THE PLATE = ";
    cin>>t;
    cout<<endl;

    cout<<"ENTER THE NUMBER OF ELEMENTS IN X-DIRECTION = ";
    cin>>Xele;
    cout<<endl;

    cout<<"ENTER THE NUMBER OF ELEMENTS IN Y-DIRECTION = ";
    cin>>Yele;
    cout<<endl;
    cout<<endl;

//  l = 7.00; w = 5.25; t = 0.125; Xele = 40; Yele = 40;

    max_node = ((Xele+1)*(Yele+1))+(Xele*Yele);

    outFile_din<<"Generated using the displacements from ABAQUS: "
        <<l<<""X"<<w<<""X"<<t<<""  plate"<<" with "<<Xele<<"X"<<Yele
```

```
                <<" mesh"<<endl;
        outFile_din<<"5"<<"   "<<max_node<<"   "<<"1   1   1   1   0   0   1"<<endl;
        outFile_din<<"0   0"<<endl<<"0"<<endl;

        double y,x;
    //  double xcor,ycor;

        x = l/Xele;
        y = w/Yele;

    //GENERATING NODAL COORDINATES FOR THE DATA INPUT FILE

        double yinc, xinc;
        yinc=0.0000000;

        while(1)
        {
                xinc = 0.0000000;
                while(1)
                {
                        outFile_din<<setw(10)<<nodeID<<setw(6)<<"0"<<setw(15)
                                        <<setprecision(8)<<fixed<<xinc<<setw(15)<<setprecision(8)
                                                <<fixed<<yinc<<setw(15)<<setprecision(8)<<fixed<<-
t/2
                                                <<setw(8)<<"0"<<setw(15)<<setprecision(8)<<fixed<<t
                                                <<setw(8)<<"0"<<endl;

                        outFile_din<<setw(31)<<setprecision(8)<<fixed<<xinc<<setw(15)

        <<setprecision(8)<<fixed<<yinc<<setw(15)<<setprecision(8)
                                                <<fixed<<t/2<<endl;
                        nodeID++;

                        if((nodeID<max_node) && (nodeID-ch == Xele+1))
                        {
                                ch = nodeID;
                                while(1)
                                {
                                        outFile_din<<setw(10)<<nodeID<<setw(6)<<"0"<<setw(15)
                        <<"0.00000000"<<setw(15)<<"0.00000000"<<setw(15)<<"0.00000000"
    <<setw(8)<<"0"<<setw(15)<<setprecision(8)<<t<<setw(8)<<"0"<<endl;

        outFile_din<<setw(31)<<"0.00000000"<<setw(15)<<"0.00000000"<<setw(15)<<"0.00000000"<
<endl;
                                        nodeID++;
                                        if(nodeID-ch == Xele)
                                        {
                                                ch = nodeID;
                                                break;
                                        }
                                }
```

```
                }
                if((nodeID % (2*Xele+1))==1 || nodeID > max_node)
                        break;
                xinc=xinc+x;
        }
        yinc=yinc+y;
        if(nodeID > max_node)
                break;
    }
/*********************************************************************/
    cout<<"THE NODAL COORDINATES ARE GENERATED"<<endl<<endl;

    outFile_din<<setw(8)<<"1"<<"       "<<"1   1   0   0   0   1"<<endl;
    outFile_din<<setw(8)<<max_node-1<<"       "<<"1   1   0   0   0   0"<<endl;

    nodeID=1;
    ch = nodeID;

    //CRAETING BOUNDARY CONDITIONS FOR THE DATA INPUT FILE
    while(1)
    {
        outFile_din<<setw(8)<<nodeID<<"     "<<" -1   -1   -1   -1   -1   0"<<endl;

        if((nodeID-ch) == Xele)
        {
                nodeID = nodeID+(Xele+1);
                ch = nodeID;
        }
        else
        {
                nodeID++;
        }
        if(nodeID > max_node)
                break;
    }
    cout<<"THE BOUNDARY CONDITIONS ARE GENERATED"<<endl<<endl;
    max_element = Xele*Yele;

    //WRITING THE MATERIAL PROPERTIES TO THE DATA INPUT FILE

    outFile_din<<"1    0    0    "<<(Xele+1)*(Yele+1)<<endl;
    outFile_din<<"4"<<setw(8)<<max_element<<"    "
        <<"2   5   3   2   1   1   0   0   0   1   0   0   0"<<endl;
    outFile_din<<"1         7.67E-04"<<endl;
    outFile_din<<"2.96E+07         0.3   0.0   0.0"<<endl;
    outFile_din<<"2         7.67E-04"<<endl;
    outFile_din<<"1.00936+07       0.3   0.0   0.0"<<endl;
    int node1, node2, node3, node4, midnode;
    node1=1;
    ch = node1;
    //GENERATING THE NODAL CONNECTIVITY
```

```
        for(int eleID = 1; eleID <= max_element; eleID++)
        {
                outFile_din<<setw(8)<<eleID<<"     5     1     0"<<endl;

                node2 = node1+1;
                node3 = node1+2*Xele+1;
                node4 = node3+1;
                midnode = node1+Xele+1;

                outFile_din<<setw(8)<<node1<<setw(8)<<node2<<setw(8)<<node4
                        <<setw(8)<<node3<<setw(8)<<midnode<<"   1     0.0"<<endl;

                node1++;
                if(node1-ch == Xele)
                {
                        node1 = node1+(Xele+1);
                        ch = node1;
                }
        }
        cout<<"ELEMENTS (NODAL CONNECTIVITY) ARE GENERATED"<<endl<<endl;

        ifstream infile2(argv[1], ios::in);
        ifstream infile3(argv[2], ios::in);

        char *out = argv[3];
        outFile_din.close();

        char *s1 = argv[1];
        getVectors(s1, out, Xele, Yele);  //CALLING THE FUNCTION FOR THE
                                  // UNDAMAGED DISPLACEMENTS
                                 //GENERATED BY THE ABAQUS
        outFile_din.open(argv[3],ios::app);
        outFile_din<<"0"<<endl;
        outFile_din.close();

        char *s2 = argv[2];
        getVectors(s2, out, Xele, Yele);  //CALLING THE FUNCTION FOR THE
                                    //DAMAGED DISPLACEMENTS
                                  //GENERATED BY THE ABAQUS
        outFile_din.open(argv[3],ios::app);
        outFile_din<<"1"<<endl;
        outFile_din<<Yele<<setw(8)<<Xele<<endl;
        outFile_din<<"0"<<endl;
        outFile_din<<"4"<<setw(8)<<"4"<<endl;

        infile2.close();
        infile3.close();
        outFile_din.close();

        return 0;
}
```

```
// FUNCTION TO GET DAMAGE FORCE VECTORS
int getVectors(char s[], char o[], int Ex, int Ey)
{
    char buffer[2093];
    char ch[5];
    double tmp;
    double freq;
    int i = 0;
    bool eof();
    char check1[] = "Value";
    char check2[] = "Node";

    int node = 1;
    int check = node;

    ifstream ifp;
    ifp.open(s, ios::in);

    ofstream ofp;
    ofp.open(o, ios::app);

    while(1)
    {
        ifp.getline(buffer, 2093,' ');
        if(strcmp(buffer,check1) == 0)
        {
            ifp>>ch;
            ifp>>freq;
            ofp<<setprecision(15)<<freq<<endl;
            break;
        }
    }
    while(1)
    {
        ifp.getline(buffer, 2093,' ');
        int r = strcmp(buffer, check2);
        if( r == 0)
        {
            ifp.getline(buffer, 2093);
            ifp.getline(buffer, 2093);
            ifp.getline(buffer, 2093);
        //WRITING THE DISPLACEMENTS TO THE DATA INPUT FILE

            while(1)
            {
                ifp>>tmp;
                ofp<<setw(8)<<node<<setw(6)<<"5"<<setw(6)<<"1";
                ifp>>tmp;
                ofp<<setw(20)<<setprecision(15)<<fixed<<tmp<<setw(6)<<2;
```

```
                                ifp>>tmp;
        ofp<<setw(20)<<setprecision(15)<<fixed<<tmp<<setw(6)<<3;
                                ifp>>tmp;
        ofp<<setw(20)<<setprecision(15)<<fixed<<tmp<<setw(6)<<4;
                                ifp>>tmp;
        ofp<<setw(20)<<setprecision(15)<<fixed<<tmp<<setw(6)<<5;
                                ifp>>tmp;
                                ofp<<setw(20)<<setprecision(15)<<fixed<<tmp<<endl;

                                if((node-check) == Ex)
                                {
                                        node = node+(Ex+1);
                                        check = node;
                                }
                                else
                                {
                                        node++;
                                }

                                //if(ifp.eof())
                                if((node-(Ex+1))==(((Ex+1)*(Ey+1))+(Ex*Ey)))
                                        break;
                                i++;
                        }
                        break;
                }
        }
        ifp.close();
        ofp.close();
        return 0;
}
```

Appendix B
Wavelet Transforms

B.1 An example of Decomposition of a Signal in Haar Wavelet

Signal, f = (4, 6, 10, 12, 8, 6, 5, 5)
Length of signal, $N = 2^3 = 8$

Averages: $a^1 = (a_1, a_2, ..., a_{N/2})$

$$a_1 = \frac{(f_1 + f_2)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_1 + f_2)}{2}\right)$$

$$a_2 = \frac{(f_3 + f_4)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_3 + f_4)}{2}\right)$$

$$a_{N/2} = \frac{(f_{N-1} + f_N)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_{N-1} + f_N)}{2}\right)$$

Fluctuations: $d^1 = (d_1, d_2, ..., d_{N/2})$

$$d_1 = \frac{(f_1 - f_2)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_1 - f_2)}{2}\right)$$

$$d_2 = \frac{(f_3 - f_4)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_3 - f_4)}{2}\right)$$

$$d_{N/2} = \frac{(f_{N-1} - f_N)}{\sqrt{2}} = \sqrt{2}\left(\frac{(f_{N-1} - f_N)}{2}\right)$$

Level-1 Decomposition: $f = (a^1 \mid d^1)$

$$f = (5\sqrt{2}, 11\sqrt{2}, 7\sqrt{2}, 5\sqrt{2} \mid -2/\sqrt{2}, -2/\sqrt{2}, 2/\sqrt{2}, 0)$$

Level-2 Decomposition: $f = (a^2 \mid d^2 \mid d^1)$

$$f = (16, 12 \mid -6, 2 \mid -2/\sqrt{2}, -2/\sqrt{2}, 2/\sqrt{2}, 0)$$

Level-N Decomposition: $f = (a^N \mid d^N \mid ..... \mid d^2 \mid d^1)$

Inverse Haar Transform from Level-1 Decomposition:

$$(a^1 \mid d^1) = \left(\frac{a_1 + d_1}{\sqrt{2}}, \frac{a_1 - d_1}{\sqrt{2}}, \frac{a_2 + d_2}{\sqrt{2}}, \frac{a_2 - d_2}{\sqrt{2}}, ..., \frac{a_{N/2} + d_{N/2}}{\sqrt{2}}, \frac{a_{N/2} - d_{N/2}}{\sqrt{2}} \right) = f$$

$$(a^1 \mid d^1) = (4,\ 6,\ 10,\ 12,\ 8,\ 6,\ 5,\ 5) = f$$

Decomposition in the form of Scaling and Wavelet Signals:
Scaling Signal:

$$V_1^1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, ..., 0 \right)$$

$$V_2^1 = \left(0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, ..., 0 \right)$$

.

.

$$V_{N/2}^1 = \left(0, 0, ...0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right)$$

$\therefore$ First Level Averages, $a_m^1 = \left\langle f, V_m^1 \right\rangle$

Wavelet Signal:

$$W_1^1 = \left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, ..., 0 \right)$$

$$W_2^1 = \left(0, 0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, ..., 0 \right)$$

.

.

$$W_{N/2}^1 = \left(0, 0, ...0, 0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right)$$

$\therefore$ First Level Fluctuations, $d_m^1 = \left\langle f, W_m^1 \right\rangle$

Hence,
Averages, $a_m^n = \left\langle f, V_m^n \right\rangle$ and Fluctuations, $d_m^n = \left\langle f, W_m^n \right\rangle$

B.2 How to create your own wavelet in wavelet toolbox in Matlab

The wavelet could be defined by some function. Be sure that the function has integral zero, so that it satisfies the requirement of being a wavelet. Equivalently, its Fourier transform should be zero at zero.

The *wavemngr* command permits to add new wavelets and wavelet families to the predefined one in the GUI. However, before you can use the *wavemngr* command to add a new wavelet, you must:

- Choose the full name of the wavelet family(fn)
- Choose the short name of the wavelet family(fsn)
- Determine the wavelet type(wt)
- Define the orders of wavelets within the given family(nums)
- Bulit a MAT-file or M-file(file)
- For wavelets without FIR filters: Define the effective support.

The short name of the wavelet family, fsn, must be a string of four characters or less. For Matlab, wavelets can be defined through the definition of the wavelet function as an m-file. To describe a wavelet, if you use type 4, write a Matlab code whose first line is function [psi, t] = file(lb, ub, n). The rest of that file should be the formula for the wavelet. For example, here is Matlab's code for the Mexican-hat wavelet (second derivative of a Gaussian function), which has interval of effective support as [-5, 5].

```
Function [ out1, out2] = Mexican(lb, ub, n)
out2 = linespace(lb, ub, n);     %WAVELET SUPPORT
out1 = out2.^2;
out1 = (2/sqrt(3)*pi^0.25))*exp(-out1/2).*(1-out1);
```

The "WAVEMNGR" command:

```
WAVEMNGR Wavelet manager.
   WAVEMNGR is a wavelet manager used to add, delete, restore or read wavelets.
   WAVEMNGR('add',FN,FSN,WT,NUMS,FILE) or
   WAVEMNGR('add',FN,FSN,WT,NUMS,FILE,B) or
   WAVEMNGR('add',FN,FSN,WT,{NUMS,TYPNUMS},FILE) or
   WAVEMNGR('add',FN,FSN,WT,{NUMS,TYPNUMS},FILE,B) adds a new wavelet family.
    FN  = family name (string).
    FSN = family short name (string).
    WT defines the wavelet type:
    WT = 1 for orthogonal wavelets.
    WT = 2 for biorthogonal wavelets.
    WT = 3 for wavelet with scale function.
    WT = 4 for wavelet without scale function.
    WT = 5 for complex wavelet without scale function.

    If the wavelet is a single one, NUMS = ''.
     examples: mexh, morl.
    If the wavelet is part of a finite family of wavelets, NUMS is a string containing a blank separated
list of items representing wavelet parameters.
     example: bior, NUMS = '1.1 1.3 ... 4.4 5.5 6.8'.
    If the wavelet is part of an infinite family of wavelets, NUMS is a string containing a blank
separated list of items representing wavelet parameters, terminated by the special sequence **.
     examples:
       db,    NUMS = '1 2 3 4 5 6 7 8 9 10 **'.
       shan,  NUMS = '1-1.5 1-1 1-0.5 1-0.1 2-3 **'
    In these last two cases, TYPNUMS specifies the wavelet parameter input format: 'integer' or 'real'
or 'string'; the default value is 'integer'.
     examples: db,   TYPNUMS = 'integer'
               bior, TYPNUMS = 'real'
               shan, TYPNUMS = 'string'
```

FILE = MAT-file or M-file name (string).
B = [lb ub] specifies lower and upper bounds of
effective support for wavelets of type = 3, 4 or 5.

WAVEMNGR('del',N), deletes a wavelet or a wavelet family where N is the wavelet name or the family short name.

WAVEMNGR('restore') restores the previous wavelets.asc ASCII-file.

WAVEMNGR('restore',IN2) restores the initial wavelets.asc ASCII-file.

OUT1 = WAVEMNGR('read') returns all wavelets family names.

OUT1 = WAVEMNGR('read',IN2) returns all wavelet names.

OUT1 = WAVEMNGR('read_asc') returns all wavelets information retrieved from wavelets.asc ASCII-file.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 2006	Final Contractor Report

4. TITLE AND SUBTITLE

Enhancement of the Feature Extraction Capability in Global Damage Detection Using Wavelet Theory

5. FUNDING NUMBERS

WBS–645846.02.07.03
NCC3–808

6. AUTHOR(S)

Atef F. Saleeb and Gopi Krishna Ponnaluru

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Akron
302 Buchtel Mall
Akron, Ohio 44325

8. PERFORMING ORGANIZATION REPORT NUMBER

E–15040

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2006-214225

11. SUPPLEMENTARY NOTES

Project manager, Steven M. Arnold, Materials and Structures Division, Glenn Research Center, organization code RXL, 216–433–3334.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 39

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The main objective of this study is to assess the specific capabilities of the defect energy parameter technique for global damage detection developed by Saleeb and coworkers. The feature extraction is the most important capability in any damage-detection technique. Features are any parameters extracted from the processed measurement data in order to enhance damage detection. The damage feature extraction capability was studied extensively by analyzing various simulation results. The practical significance in structural health monitoring is that the detection at early stages of small-size defects is always desirable. The amount of changes in the structure's response due to these small defects was determined to show the needed level of accuracy in the experimental methods. The arrangement of fine/extensive sensor network to measure required data for the detection is an "unlimited" ability, but there is a difficulty to place extensive number of sensors on a structure. Therefore, an investigation was conducted using the measurements of coarse sensor network. The white and the pink noises, which cover most of the frequency ranges that are typically encountered in the many measuring devices used (e.g., accelerometers, strain gauges, etc.) are added to the displacements to investigate the effect of noisy measurements in the detection technique. The noisy displacements and the noisy damage parameter values are used to study the signal feature reconstruction using wavelets. The enhancement of the feature extraction capability was successfully achieved by the wavelet theory.

14. SUBJECT TERMS

Experimentation; Modal testing; Detection; Health monitoring; Damage

15. NUMBER OF PAGES

148

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	